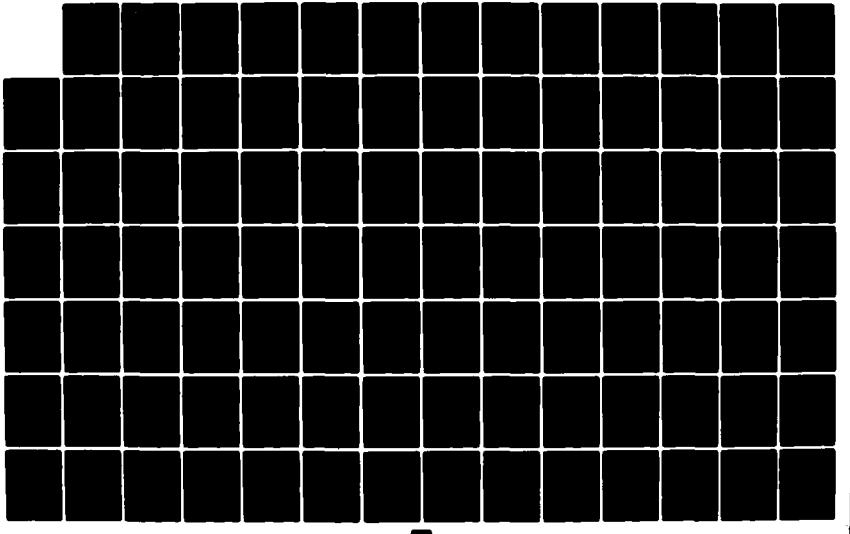
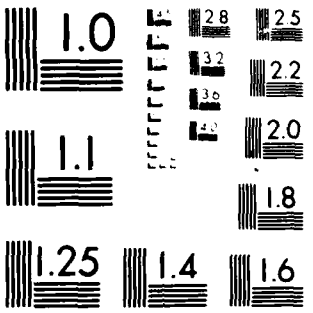
AD-A124 064 AN INVESTIGATION OF ORDERING TEARING AND LATENCY ALGORITHMS FOR THE TIME-..(U) ILLINOIS UNIV AT URBANA COORDINATED SCIENCE LAB P YANG AUG 80 R-891 1/2

UNCLASSIFIED N00014-79-C-0424 F/G 9/5 NL





REPORT R-891 AUGUST, 1980 UILU-ENG 80-2223

COORDINATED SCIENCE LABORATORY

AN INVESTIGATION OF ORDERING, TEARING, AND LATENCY ALGORITHMS FOR THE TIME-DOMAIN SIMULATION OF LARGE CIRCUITS

PING YANG

UNIVERSITY OF ILLINOIS - URBANA, ILLINOIS

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A124064	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) AN INVESTIGATION OF ORDERING, TEARING, AND LATENCY ALGORITHMS FOR THE TIME-DOMAIN SIMULATION OF LARGE CIRCUITS		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER R-880; UILU-ENG 80-2212
7. AUTHOR(s) Ping Yang		8. CONTRACT OR GRANT NUMBER(s) N00014-79-C-0424
9. PERFORMING ORGANIZATION NAME AND ADDRESS Coordinated Science Laboratory University of Illinois at Urbana-Champaign Urbana, Illinois 61801		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Joint Services Electronics Program		12. REPORT DATE August 1980
		13. NUMBER OF PAGES 176
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Integrated Circuits
Computer-Aided Analysis Program
DC and Transient Analysis Including Tearing and Latency

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Many circuit simulation programs have been available for the design of integrated circuits. However, these conventional circuit simulation programs calculate all of the node voltages or branch voltages and currents at each iteration and each timepoint. Even with sparse matrix techniques the simulation of modern large-scale integrated (LSI) circuits is not possible in many situations due to the excessive computation time and high storage requirements.

The goal of this research was to investigate new approaches to the

DD FORM 1 JAN 73 1473

UNCLASSIFIED

(Over)

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

simulation of integrated circuits which can alleviate the problems of excessive computation time and high storage requirements. A new ordering scheme for the modified nodal approach was developed, and some new algorithms for the dc and transient analysis of logic circuits were studied. Different tearing methods and sparsity considerations for the node tearing method were theoretically and experimentally studied. Latency at the subcircuit and the network levels was investigated. Different latency criteria were proposed and studied. The result of this research is a new general purpose circuit simulation program SLATE.

AN INVESTIGATION OF ORDERING, TEARING, AND LATENCY ALGORITHMS FOR THE TIME-DOMAIN SIMULATION OF LARGE CIRCUITS

by

Ping Yang

This work was supported by the Joint Services Electronics Program under Contract N00014-79-C-0424.

Approved for public release. Distribution unlimited.

AN INVESTIGATION OF ORDERING, TEARING, AND LATENCY ALGORITHMS FOR THE TIME-DOMAIN SIMULATION OF LARGE CIRCUITS

BY

PING YANG

B.S., National Taiwan University, 1974
M.S., University of Illinois, 1978

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1980

Thesis Adviser: Professor Timothy N. Trick

and

Professor Ibrahim N. Hajj

Urbana, Illinois

AN INVESTIGATION OF ORDERING, TEARING, AND LATENCY ALGORITHMS FOR THE TIME-DOMAIN SIMULATION OF LARGE CIRCUITS

Ping Yang, Ph.D.

Coordinated Science Laboratory and
Department of Electrical Engineering
University of Illinois at Urbana-Champaign, 1979

Many circuit simulation programs have been available for the design of integrated circuits. However, these conventional circuit simulation programs calculate all of the node voltages or branch voltages and currents at each iteration and each timepoint. Even with sparse matrix techniques the simulation of modern large-scale integrated (LSI) circuits is not possible in many situations due to the excessive computation time and high storage requirements.

The goal of this research was to investigate new approaches to the simulation of integrated circuits which can alleviate the problems of excessive computation time and high storage requirments. A new ordering scheme for the modified nodal approach was developed, and some new algorithms for the dc and transient analysis of logic circuits were studied. Different tearing methods and sparsity considerations for the node tearing method were theoretically and experimentally studied. Latency at the subcircuit and the network levels was investigated. Different latency criteria were proposed and studied. The result of this research is a new general purpose circuit simulation program SLATE.

ACKNOWLEDGEMENTS

First, I would like to thank Professor T. N. Trick, my dissertation advisor. With his thorough knowledge of the field, he provided valuable guidance and help throughout the course of this research. Also, with his great personality, he provided support and encouragement in all respects of my life throughout my graduate study. I would also like to thank Professor I. N. Hajj, my dissertation co-advisor, for the many helpful discussions with him and suggestions. I am also grateful to Professor M. R. Lightner, who invented the name 'SLATE' for my program and offered a lot of advice and encouragement. I also wish to thank Professor W. R. Perkins for being member of my dissertation committee and his support.

I am also indebted to my wife, Jau-Yuann, for her help in preparing this dissertation and her understanding throughout my graduate study.

Finally, I would like to thank my parents, I-Lueh and Queh-Chu Yang, for their love and support throughout the past years.

TABLE OF CONTENTS

I. INTRODUCTION

The design of integrated circuits requires an accurate method of predicting circuit performance. The traditional breadboard method is not able to satisfy the above requirement because of the fact that the parasitic components that are present in the breadboard are entirely different from the parasitic components that are present in integrated circuits, so a circuit simulation program is a must. Conventional circuit simulation programs [1-11] possess two serious limitations: a computer storage requirement and a computing time requirement, so the size of the circuit that can be simulated is limited. With the advances of circuit simulation techniques, the size of the circuit that can be simulated has increased; but the simulation of large scale integrated (LSI) circuits is still beyond the capabilities of present circuit simulation programs.

The goal of this research was to study new approaches to the simulation of integrated circuits which can alleviate the above two limitations, namely the repetitiveness and latency properties of digital integrated circuits. Since a DEC-10 version of SPICE2 was available to us, it was decided that this program would serve as a vehicle for testing our algorithms. However, in the initial phases of our research, it was found that our version of SPICE2 had several deficiencies in the implementation of some of its algorithms which occasionally caused numerical difficulties. In order to resolve these difficulties a new reordering scheme for the modified nodal approach was developed, a new concept - a piecewise nonlinear approach

- for the Newton-Raphson iteration was proposed, and two problems with the numerical integration algorithm were resolved. The new reordering scheme for the modified nodal approach not only avoids zero diagonal pivot elements which increases numerical accuracy, but it also significantly reduces the number of fills in the matrix which reduces the computational cost. The piecewise nonlinear approach reduces the number of iterations needed to find the solution of a nonlinear circuit and improves the global convergence property of Newton-Raphson method. The resolution of the two problems with the numerical integration algorithm provides more efficiency and accuracy. All of these new developments result in a modified version of SPICE2 (YSPICE), which is 2 to 5 times faster than SPICE2.

Although YSPICE is more efficient and more accurate than SPICE2, it is still not powerful enough to handle LSI circuits simulation problems. Experience has shown that LSI circuits possess properites which can be exploited to improve the storage and computing time requirements. The two properties are the repetitiveness of a limited number of subcircuits and the latency that may exist within parts of the circuits during an analysis. Conventional circuit simulation programs do not exploit these two properties, so all of the node voltages or branch voltages and currents are calculated at each iteration and each timepoint. In order to increase the capabilities of circuit simulation programs substantially, these two properties must be fully exploited. When the first property is exploited both computer storage requirements and computing time can be reduced in several ways.

First, only one subcircuit description for each type of repetitive subcircuit need be stored; secondly, only one set of small submatrix sparse matrix pointers for each type of repetitive subcircuit is needed so that both storage and preprocessing time can be saved; thirdly, if one type of subcircuit is linear, then the LU factorization of that type of subcircuit need be found once only. When the second property is exploited, we only need to solve for the active parts of the circuit and this reduces the computational effort considerably. Tearing methods, first introduced by Kron [12], are well suited for the exploitation of these two properites as well as the sparsity of the network. Recently, the use of tearing methods and latency [13-24] has been studied to exploit these two properties, but in order to fully exploit these two properties more research effort is needed.

In the second stage of our research, these two problems were studied extensively and the result of our investigations is a new general purpose circuit simulation program SLATE (a Simulator with Latency and Tearing). SLATE evolved from YSPICE, so it has all the good features of YSPICE: in addition, several new approaches are used. First, the new reordering strategy for the modified nodal approach is used at both the subcircuit and interconnection levels; secondly, ways of exploiting sparsity that exist at the subcircuit and interconnection levels were theoretically and experimentally studied and the most efficient way is used; thirdly, node tearing is used such that the program is more efficient and the final equation formulation is suitable for latency exploitation and parallel processing; fourthly,

latency in he Newton-Raphson iterations is exploited not only at device and subcircuit levels, but also at interconnection levels; fifthly, latency in the time domain is exploited not only at device and subcircuit levels, but also at interconnection levels; sixthly, three latency in time criteria schemes were studied thoroughly in relation to the spread of the time constants in the subcircuits and the best scheme was determined; and lastly, the interconnection matrix formulation method is general enough to accommodate the situation when there are no subcircuits specified in the network or when the interconnection circuits consist of more than tearing nodes.

Both YSPICE and SLATE are written in FORTRAN and have a SPICE-like input language for user convenience. If no subcircuits are used, then the methods of analysis of SLATE is equivalent to that of YSPICE, that is, YSPICE is a subset of this new program SLATE. Simulation results indicate that the speed of SLATE is about an order of magnitude faster than SPICE2, and the output results are either the same as or more accurate than those of SPICE2.

The new reordering scheme for the modified nodal approach is described in Chapter 2, and the comparison between this new scheme and that used in SPICE2 is given The piecewise nonlinear approach is explained in Chapter 3 and simulation results are given. The two problems with numerical integration are detailed in Chapter 4, and the solution is given. Chapter 5 introduces the concept of tearing methods and gives the sparsity consideration for the node tearing method. Chapter 6 describes three latency criteria and gives the simulation

results of these three schemes. Finally, in Chapter 7 a summary of SLATE performance is given, the conclusions are presented, and areas for future work are described.

II. NEW REORDERING STRATEGY FOR THE MODIFIED NODAL APPROACH

The modified nodal approach (MNA) [25] has been widely used in many computer-aided circuit analysis programs [1,11,26,27] for formulating circuit equations. It is well known, however, that while the more restrictive nodal approach in general produces nonzero diagonal elements for pivoting, the modified nodal approach, although more general, may produce zero diagonal entries in the network matrix. This occurs, for example, when the circuit contains voltage sources, short-circuits, inductors at zero frequency (dc solution) and some types of controlled sources. When sparse matrix techniques with diagonal pivoting are used for solving these types of circuit equations, extreme care should be taken so as not to choose a zero-valued pivot. Two methods have been proposed for avoiding pivoting on these zero diagonal entries. One method (method 1) involves ordering the rows and columns with zero diagonal entries last, in the hope that they will be filled before becoming candidates for pivoting [1,11]. Another method (method 2) involves rearranging and/or combining rows and columns in order to obtain nonzero diagonal elements [25]. However, as we show below, there are two problems with these methods. First, even if all the zero diagonal elements which exist in the network matrix at the formulation stage are avoided or filled during the elimination stage, it is possible to generate zero diagonal elements during the Gaussian elimination process regardless of the values of the circuit elements; Secondly, these methods usually are not efficient. For example, forcing the zero-diagonal entries to be last usually increases the number of fills considerably.

In this chapter a new reordering scheme for the modified nodal approach is described which avoids zero diagonal pivots in essentially all practical cases and is very efficient. In Section 2.1, the problems with previous methods are illustrated and explained. In Section 2.2, the partitioning of the circuit variables is detailed and the ordering strategy is introduced. In Section 2.3, theorems and examples are given. The implementation of this new scheme resulted in YSPICE. The simulation results from YSPICE are given in Section 2.4. In this Section examples are given which caused computational problems in our DEC-10 version of SPICE2 due to pivoting on zero diagonal elements, but which were successfully analyzed by YSPICE. Also the number of fills produced by YSPICE is much less than that produced by SPICE2. In Section 2.5, a discussion of this new ordering strategy is given.

2.1. Problems with Previous Methods

The MNA matrix can in general be expressed in the form [25]

$$\begin{bmatrix} \underset{\sim}{Y}_R & \underset{\sim}{B} \\ \underset{\sim}{C} & \underset{\sim}{D} \end{bmatrix} \begin{bmatrix} \underset{\sim}{V} \\ \underset{\sim}{I} \end{bmatrix} = \begin{bmatrix} \underset{\sim}{J} \\ \underset{\sim}{E} \end{bmatrix} \tag{2.1}$$

where $\underset{\sim}{V}$ is the set of node-to-datum voltages and $\underset{\sim}{I}$ is the set of branch currents which are chosen as additional circuit variables. $\underset{\sim}{Y}_R$ is a reduced form of the nodal matrix excluding the contributions due to voltage sources, current controlling elements, etc. $\underset{\sim}{B}$ contains partial derivatives of the Kirchhoff current equations with respect to the additional current variables and thus contains ± 1's for the elements whose branch relations

are introduced. The branch constitution relations, differentiated with respect to the unknown vector are represented by the matrices $\underset{\sim}{C}$ and $\underset{\sim}{D}$. $\underset{\sim}{J}$ and $\underset{\sim}{E}$ are the excitations.

As mentioned above, when sparse matrix techniques with diagonal pivoting are used for solving Eq. (2.1), zero diagonal elements may be encountered. Previously, two methods have been proposed for avoiding pivoting on these zero diagonal elements. However, there are still two problems with these previous methods: (1) zero diagonal elements may be generated during the Gaussian elimination process, and (2) the methods may not be the most efficient. In this section we consider the zero diagonal problem, and in Section 2.4 we discuss the efficiency problem.

Method 1 orders the rows and columns with zero diagonal entries last, in the hope that they will be filled before becoming candidates for pivoting. Even if all the zero diagonal elements which exist in the network matrix at the formulation stage are filled during the elimination stage, cutsets of branches whose currents are declared as network variables in a modified nodal formulation will generate zero diagonal elements during the Gaussian elimination process regardless of the values of the circuit elements. This problem is proved and illustrated by Theorem 2.1, Example 2.1, and Example 2.2.

Theorem 2.1. For any network which has cutsets of branches whose currents are declared as circuit variables in a modified nodal formulation, if these current variables are ordered last, then zero diagonal elements will be generated during the Gaussian elimination process, regardless of circuit element values.

Proof: Since we assume that all the current variables are ordered last and they form cutsets, therefore floating subnetworks are created. The admittance matrices of these floating subnetworks are singular, therefore the $\underset{\sim}{Y}_R$ in Eq. (2.1) is singular, so zero diagonal elements will be generated during the Gaussian elimination of Eq. (2.1).

The following Example 2.1 illustrates Theorem 2.1.

Example 2.1: A cutset of current variables (Fig. 2.1)

If the method which orders all the current variables last is used to formulate the modified nodal equations of the circuit shown in Fig. 2.1, the resulting equations will be as follows:

$$\begin{bmatrix} G_1 & -G_1 & 0 & 1 & 0 \\ -G_1 & G_1 & 0 & 0 & -1 \\ 0 & 0 & G_2 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ I_E \\ I_L \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ E \\ 0 \end{bmatrix}$$

During the course of Gaussian elimination due to the resulting floating subnetwork, a zero diagonal element will be produced at location (2,2).

Remark: For any subnetwork which has cutsets of branches whose currents are declared as circuit variables in a modified nodal formulation, if the rows corresponding to current variables which have zero-diagonal elements

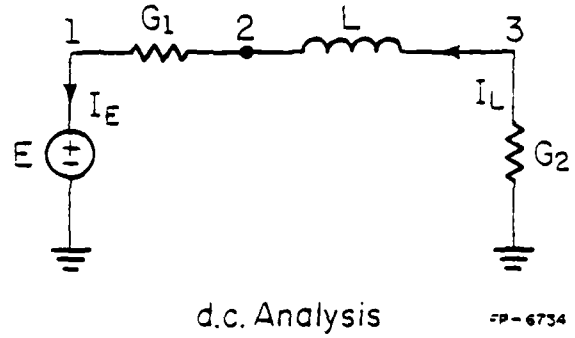


Fig. 2.1 Circuit used in Example 2.1.

are ordered last until a diagonal entry is filled, before it is considered as a pivot, then zero diagonal elements may be generated during the Gaussian elimination process, regardless of circuit element values.

The proof of this remark is the same as that of Theorem 2.1. In the following, Example 2.2 illustrates this remark.

Example 2.2: A cutset of current variables (Fig. 2.2)

If the reordering strategy mentioned in the previous remark is used to formulate the equations of the circuit shown in Fig. 2.2, the matrix formulated is:

$$\begin{bmatrix} G_2 & 0 & 0 & 0 \\ 0 & G_1 & -G_1 & 1 \\ 0 & -G_1 & G_1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_3 \\ V_1 \\ V_2 \\ I_E \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ E \end{bmatrix}$$

During the course of Gaussian elimination due to the resulting floating subnetwork, a zero diagonal element will be generated at location (3,3).

Method 2 interchanges rows in order to obtain nonzero diagonal elements. Even if all the zero diagonal elements which exist in the network matrix at the formulation stage are avoided before the elimination, if there are loops of branches whose currents are declared as network variables in the modified nodal formulation, then zero diagonal elements

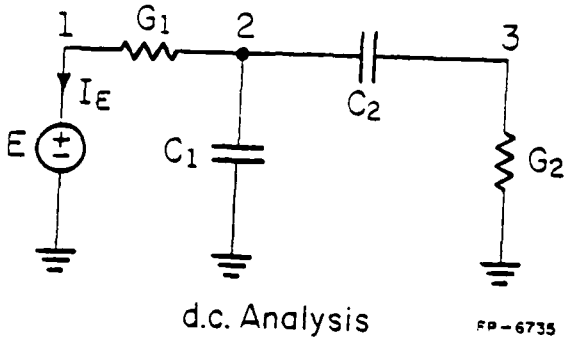


Fig. 2.2 Circuit used in Example 2.2.

may be generated during the Gaussian elimination process regardless of the values of the circuit elements. This problem is proved and illustrated by Theorem 2.2 and Example 2.3.

Let us define the branch whose current is declared as a current variable in the modified nodal formulation as current branch. Let us define the 'positive' node as follows: Assuming that the datum node can not be chosen as 'positive' and that the datum node is not contained in any loop formed by current branches, then we can always choose one of the two nodes of a current branch as 'positive' for that current branch and there is a one-to-one correspondence between these 'positive' nodes and the current branches. An algorithm for choosing 'positive' nodes is given in Section 2.2.

Theorem 2.2. For any network with a loop of branches whose currents are declared as network variables in a modified nodal formulation, and the reference node is not contained in the loop and there is no coupling among the voltages of the branches in the loop, then if all the rows corresponding to the current variables are interchanged with the corresponding 'positive' node voltage rows, zero diagonal elements will be generated during the Gaussian elimination process, regardless of circuit element values.

Proof: Let us assume that after the rows corresponding to the current variables are interchanged with the corresponding 'positive' node voltage rows, the rows corresponding to the current variables are ordered first, then the MNA matrix equation (2.1) is transformed into

$$\begin{pmatrix} \underset{\sim}{B}_1 & \underset{\sim}{Y}_{12} & \underset{\sim}{Y}_{11} \\ \underset{\sim}{B}_2 & \underset{\sim}{Y}_{22} & \underset{\sim}{Y}_{21} \\ \underset{\sim}{D} & \underset{\sim}{C}_2 & \underset{\sim}{C}_1 \end{pmatrix} \begin{pmatrix} \underset{\sim}{I} \\ \underset{\sim}{V}_2 \\ \underset{\sim}{V}_1 \end{pmatrix} = \begin{pmatrix} \underset{\sim}{J}_1 \\ \underset{\sim}{J}_2 \\ \underset{\sim}{E} \end{pmatrix} \tag{2.2}$$

The submatrix being eliminated first is the node-to-branch incidence matrix for the 'positive' nodes and the current variable branches [28], that is , the $\underset{\sim}{B}_1$ in Eq. (2.2). Since we assume that the reference node is not contained in the loop and there is no coupling among the voltages of the branches of the loop, then there is a one-to-one correspondence between each branch of the loop and the corresponding 'positive' node and each column in $\underset{\sim}{B}_1$ contains exactly a +1 and a -1, therefore, $\underset{\sim}{B}_1$ is singular and zero diagonal elements will be generated during the Gaussian elimination.

The following Example 2.3 illustrates Theorem 2.2.

Example 2.3: A loop of current variables (Fig. 2.3)

THe circuit equations formulated by method 1 for the circuit shown in Fig. 2.3 in a transient analysis using a backward Euler Formula with timestep h have the following form:

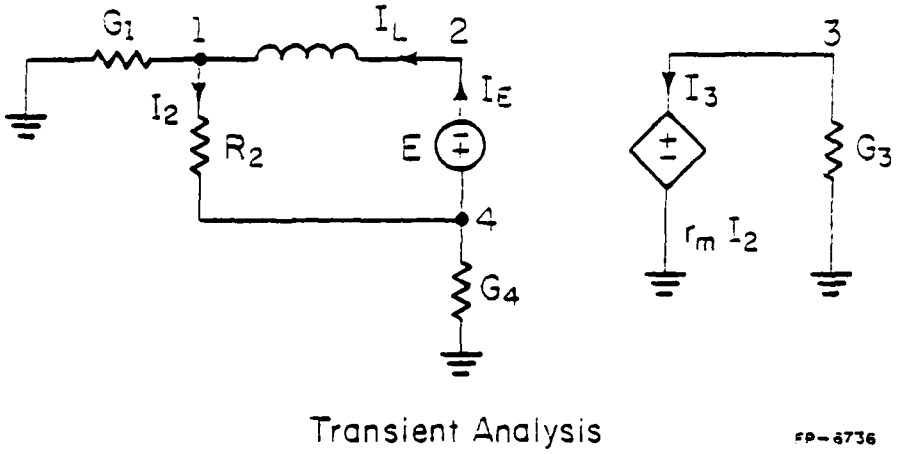


Fig. 2.3 Circuit used in Example 2.3.

$$\begin{bmatrix} 1 & 0 & 0 & -1 & G_4 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & G_3 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & G_2 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & \frac{-1}{R_2} & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & -V_m & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -R_1 & -1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_2 \\ I_1 \\ I_3 \\ I_4 \\ V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ E \\ E_1 \\ 0 \\ 0 \end{bmatrix}$$

The submatrix B_1 is singular, therefore during the Gaussian elimination a zero diagonal element will be generated at location (4,4).

2.2. New Partitioning and Ordering Strategy

From the previous section, we conclude that the topological reasons for zero diagonal elements being generated in the modified nodal approach are: (1) cutsets of current variables and (2) loops of current variables. Here we present a new partitioning and ordering strategy which has the following good features:

(1) zero diagonal elements are avoided before the Gaussian elimination and during the Gaussian elimination in essentially all practical circuits;

(2) it is efficient and the number of fills is less than that of previous

methods;

(3) it is easy to implement and the partitioning and ordering are done in the preprocessing phase, so it is well suited for the use of sparse matrix techniques.

Consider a linear (or linearized) circuit which contains independent current and voltage sources, two terminal resistors, capacitors, inductors and all types of controlled sources. We assume that the circuit contains neither loops of only (independent and dependent) voltage sources and inductors nor cutsets of only (independent and dependent) current sources and capacitors.

In the modified nodal approach, the circuit variables consist of node-to-datum voltage $\underset{\sim}{V}_n$ together with a subset of branch currents $\underset{\sim}{I}_b$. (Henceforth those branches are referred to as current branches.) In the proposed ordering strategy, the node voltages $\underset{\sim}{V}_n$ are partitioned into two subsets, $\underset{\sim}{V}_1$ and $\underset{\sim}{V}_2$, and $\underset{\sim}{I}_b$ is partitioned into three subsets, $\underset{\sim}{I}_1$, $\underset{\sim}{I}_2$ and $\underset{\sim}{I}_3$. The components of $\underset{\sim}{I}_1$ consist of the currents in the (dependent and independent) voltage sources, and are in turn partitioned as follows:

$\underset{\sim}{I}_V$ ≡branch currents of the independent voltage sources.

$\underset{\sim}{I}_{VCV}$ ≡branch currents of the voltage-controlled voltage sources.

$\underset{\sim}{I}_{CCV}$ ≡branch currents of the current-controlled voltage sources.

The components of $\underset{\sim}{I}_2$ and $\underset{\sim}{I}_3$ consist of the remaining currents which are circuit variables.

Let a graph $\underset{\sim}{G}_I$ (possibly disconnected) be first constructed to include all the current branches, with all the other branches removed. If $\underset{\sim}{G}_I$ contains loops, then a tree (or forest) is chosen, with only finite-valued resistors as links. This is always possible since by assumption no loops of only voltage sources, inductors and zero-valued resistors exist in the circuit. Let $\underset{\sim}{I}_3$ be the set of currents in the links of $\underset{\sim}{G}_I$, then these links can not form cutsets [28]. The components of $\underset{\sim}{I}_2$ consist of currents in the inductors and the remaining currents of the current resistors.

The components of $\underset{\sim}{V}_1$ consist of the following:

$\underset{\sim}{V}_V$ ≡set of 'positive' node voltages of the independent voltage sources.

$\underset{\sim}{V}_{VCV}$ ≡set of 'positive' node voltages of the voltage-controlled voltage sources.

$\underset{\sim}{V}_{CCV}$ ≡set of 'positive' node voltages of the current-controlled voltage sources.

$\underset{\sim}{V}_{bC}$ ≡set of 'positive' node voltages of the the $\underset{\sim}{I}_2$ branches.

The components of $\underset{\sim}{V}_2$ consist of the remaining node voltages.

The following algorithm is followed in selecting the 'positive' node voltages defined above:

Algorithm

(1) The ungrounded nodes of all grounded current branches belonging to $\underset{\sim}{I}_1$ or $\underset{\sim}{I}_2$ are chosen first as 'positive';

(2) Let b_j be the number of branches whose currents belong to $\underset{\sim}{I}_1$ or $\underset{\sim}{I}_2$ and which are incident at node j. Whenever a node of a current branch is chosen as 'positive', the number b_k at its 'negative' node k is reduced by one.

(3) If the b_k value of node k of a current branch is one and that node has not been previously selected as 'positive', then node k is selected 'positive' for that particular current branch. If more than one node have their b_k value equal to one and if some of these nodes do not have a conductance (i.e., a resistance whose current is not a circuit variable) connected to them, then one of these nodes is chosen 'positive' first. Otherwise, any one of the nodes that has its b_k value equal to one is chosen 'positive'.

Step (2) and (3) are repeated until all the branches corresponding to $\underset{\sim}{I}_1$ and $\underset{\sim}{I}_2$ have been processed. Note that up to this point there is always at least one node whose b_k value is one. This is because $\underset{\sim}{I}_1$ and $\underset{\sim}{I}_2$ do not form loops. Note also that the number of positive nodes is equal to the number of elements in $\underset{\sim}{I}_1$ and $\underset{\sim}{I}_2$. The polarities of the currents in the current branches are associated with the positive node assignments.

Partitioning and ordering the circuit variables in the order of $\underset{\sim}{I}_1$, $\underset{\sim}{I}_2$, $\underset{\sim}{V}_1$, $\underset{\sim}{V}_2$, and $\underset{\sim}{I}_3$, and writing the modified nodal equations in the usual way [25], we get the following equation structure:

$$\begin{bmatrix} \underset{\sim}{0} & \underset{\sim}{1}_1 & \underset{\sim}{A}_3 & \underset{\sim}{0} \\ \underset{\sim}{B}_1 & \underset{\sim}{B}_2 \; \underset{\sim}{B}_3 & \underset{\sim}{B}_4 & \underset{\sim}{B}_5 \\ \underset{\sim}{I}_1 \; \underset{\sim}{A}_2 \; \underset{\sim}{A}_1 \; \underset{\sim}{1}_1 & \underset{\sim}{Y}_{11} & \underset{\sim}{Y}_{12} & \underset{\sim}{A}_4 \\ \underset{\sim}{A}_5 & \underset{\sim}{Y}_{21} & \underset{\sim}{Y}_{22} & \underset{\sim}{A}_6 \\ \underset{\sim}{0} & \underset{\sim}{A}_7 & \underset{\sim}{A}_8 & \underset{\sim}{Z} \end{bmatrix} \begin{bmatrix} \underset{\sim}{I}_V \\ \underset{\sim}{I}_{VCV} \\ \underset{\sim}{I}_{CCV} \\ \underset{\sim}{I}_2 \\ \underset{\sim}{V}_V \\ \underset{\sim}{V}_{VCV} \\ \underset{\sim}{V}_{CCV} \\ \underset{\sim}{V}_{bC} \\ \underset{\sim}{V}_2 \\ \underset{\sim}{I}_3 \end{bmatrix} = \begin{bmatrix} \underset{\sim}{E}_V \\ \underset{\sim}{E}_{VCV} \\ \underset{\sim}{E}_{CCV} \\ \underset{\sim}{E}_2 \\ \\ \underset{\sim}{J}_1 \\ \\ \\ \underset{\sim}{J}_2 \\ \underset{\sim}{E}_3 \end{bmatrix} \tag{2.3}$$

Where the $\underset{\sim}{A}_i$'s contain the partial derivatives of the Kirchhoff current equations with respect to the circuit current variables, $\underset{\sim}{I}_1$, $\underset{\sim}{I}_2$, $\underset{\sim}{I}_3$, and thus contain 0, +1, -1 only.

By interchanging the rows corresponding to $\underset{\sim}{V}_1$ with the rows corresponding to $\underset{\sim}{I}_1$ and $\underset{\sim}{I}_2$, Eq. (2.3) can be written in the following form (This interchange is equivalent to off-diagonal pivoting and is done in practice by a simple change in the pointer system rather than a physical interchange of data in the rows.)

$$
\left[\begin{array}{c|c|cc}
\begin{matrix} 1_1 & & & \\ & \ddots & & \\ & & \ddots & A_2 \\ A_1 & & & 1_1 \end{matrix} & Y_{11} & Y_{12} & A_4 \\
\hline
0 & \begin{matrix} 1_1 & & A_3 \\ & \ddots & \\ & & \ddots \end{matrix} & & 0 \\
\hline
B_1 & \begin{matrix} B_2 & & B_3 \\ & \ddots & \\ & & 1_1 \end{matrix} & B_4 & B_5 \\
\hline
A_5 & Y_{21} & Y_{22} & A_6 \\
\hline
0 & A_7 & A_8 & Z
\end{array}\right]
\left[\begin{array}{c} I_V \\ I_{VCV} \\ I_{CCV} \\ I_2 \\ \hline V_V \\ V_{VCV} \\ V_{CCV} \\ V_{bC} \\ \hline V_2 \\ \hline I_3 \end{array}\right]
=
\left[\begin{array}{c} J_1 \\ \hline E_V \\ E_{VCV} \\ E_{CCV} \\ E_{bC} \\ \hline J_2 \\ \hline E_3 \end{array}\right]
\qquad (2.4)
$$

The circuit variables are partitioned into three subgroups: (1) I_1 and I_2, (2) V_V, and (3) the remaining variables. The Markowitz scheme [29] is used to minimize the number of operations within each subgroup. After reordering, Eq. (2.4) can now be solved by Gaussian elimination or LU factorization.

2.3 Theorems and Examples

If there are no current-controlled current sources or if the current-controlled current sources are not incident at the 'positive' nodes, then within the first two subgroups all the diagonal elements remain 1's and all the nonzero off-diagonal elements are -1's during the Gaussian elimination process, so the leading part of the elimination can be done simply by addition. The proof is given below in Theorem 2.3. Let us consider the first subgroup, the submatrix associated is the node-to-branch incidence matrix A_a for the 'positive' nodes and the currents belong to I_1 and I_2. Let us denote the directed graph of those nodes and currents by

G_I. Due to our partitioning and ordering strategy, there are no loops in G_I, so $\underset{\sim}{A}_a$ has +1's on the diagonal, 0's or -1's on the off-diagonal and $\underset{\sim}{A}_a$ is square and nonsingular.

Theorem 2.3. For any diagonal pivoting the LU factors of $\underset{\sim}{A}_a$ have the following special properties: all the diagonal elements remain +1's and all the nonzero off-diagonal elements are -1's.

Proof: Let $\underset{\sim}{A}_a$ be formulated with current I_k chosen as the first pivot where I_k flows in branch b_k, which is connected between node i and node j. After row and column interchange the first row and column of $\underset{\sim}{A}_a$ will have the following form:

$$\underset{\sim}{A}_a = \begin{array}{c} \\ 1 \\ \\ \\ \\ \\ j \\ \\ \\ \\ n \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & . & . & . & . & j & . \quad . \quad n \end{array} \\ \left[\begin{array}{c:cccccc} 1 & a_{12} & & & 0 & & a_{1n} \\ \hdashline & 1 & & & & & \\ & & \ddots & & & & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ -1 & & & & & 1 & \\ & & & & & & \ddots \\ & & & & & & & 1 \end{array}\right] \end{array}$$

where node j is assumed to be in G_I; otherwise column one would be all zeros below the diagonal. Note that the entry $a_{1j} = 0$ because G_I does not have any loops and a_{1i} i=2,3,....,n are either zero or -1. Pivoting on a_{11} amounts simply to adding row 1 to row j. Since adding any two rows in the incidence matrix of a directed graph produces a row with 0, -1 or +1

entries, row j will then contain 0 and -1's with +1 on the diagonal because $a_{1j} = 0$.

Let the submatrix generated by pivoting on a_{11} be denoted by $\hat{\underset{\sim}{A}}_a$. $\hat{\underset{\sim}{A}}_a$ can be considered as the incidence matrix of a directed graph $\hat{G}_I$ where $\hat{G}_I$ is derived from G_I by removing branch b_k and merging node 1 with node j. Thus $\hat{\underset{\sim}{A}}_a$ has the same properties as $\underset{\sim}{A}_a$, and pivoting on its first diagonal entry will produce a submatrix with ones on the diagonal and 0 and -1's elsewhere. This proves the theorem.

The reasoning for the second subgroup is similar to Theorem 2.3.

Now we would like to present the main result.

Main Result. For any network which has a unique solution, if the partitioning and orderinng strategy proposed here is used to solve the modified nodal equations, then no zero diagonal elements will be encountered during the Gaussian elimination process, except for the case when controlled sources or negative-valued elements with some specific set of circuit element values result in perfect cancellation.

Proof: There are two kinds of zero diagonal elements which may be encountered. One type is due to the formulation method [25] and occurs in the network matrix before the elimination process starts. These zero diagonal elements are avoided by interchanging the rows corresponding to the 'positive' node voltages with the rows corresponding to $\underset{\sim}{I}_1$ and $\underset{\sim}{I}_2$. During the elimination, topologically, the zero diagonal elements are caused either by ordering loops of current variables first or by a floating subnetwork which results by ordering a cutset of current variables last.

Both of these situations are prevented by partitioning $\underset{\sim}{I}_3$ away from $\underset{\sim}{I}_2$ and ordering $\underset{\sim}{I}_1$ and $\underset{\sim}{I}_2$ first, so no loops can be formed by $\underset{\sim}{I}_1$ and $\underset{\sim}{I}_2$. Since $\underset{\sim}{I}_3$ consists of currents in the links, so $\underset{\sim}{I}_3$ will not form cutsets, and thus no floating subnetworks will result.

Alternatively, this theorem can be proved as follows: Since all the currents are ordered first and eliminated first, from Theorem 2.3, we know that the elimination of these currents will not generate zero diagonal elements. After all these currents are eliminated, if $\underset{\sim}{I}_3$ is empty, we are left with nodal matrix equations, then no zero diagonal elements will be generated; if $\underset{\sim}{I}_3$ is not empty, since $\underset{\sim}{I}_3$ can not form loops or cutsets, so no zero diagonal elements will be generated.

A more rigorous and general proof can be found in [30].

In the following we would like to use the new ordering scheme to solve those examples used in Section 2.1.

Example 2.1:

If our approach is used, initially, the matrix formulated is:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & G_1 & 0 & -G_1 \\ 0 & 1 & 0 & 0 & G_2 \\ 0 & -1 & -G_1 & 0 & G_1 \end{bmatrix} \begin{bmatrix} I_E \\ I_1 \\ V_1 \\ V_3 \\ V_2 \end{bmatrix} = \begin{bmatrix} E \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

After interchanging rows, the resulting matrix is:

$$\begin{bmatrix} 1 & 0 & G_1 & 0 & -G_1 \\ 0 & 1 & 0 & 0 & G_2 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & -1 & -G_1 & 0 & G_1 \end{bmatrix} \begin{bmatrix} I_E \\ I_L \\ V_1 \\ V_3 \\ V_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ E \\ 0 \\ 0 \end{bmatrix}$$

No zero diagonal elements will be encountered during the course of Gaussian elimination.

Example 2.2:

If our approach is used, initially, the matrix formulated is:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & G_1 & -G_1 & 0 \\ 0 & -G_1 & G_1 & 0 \\ 0 & 0 & 0 & G_2 \end{bmatrix} \begin{bmatrix} I_E \\ V_1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} E \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

After interchanging rows, the resulting matrix is:

$$\begin{bmatrix} 1 & G_1 & -G_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -G_1 & G_1 & 0 \\ 0 & 0 & 0 & G_2 \end{bmatrix} \begin{bmatrix} I_E \\ V_1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} 0 \\ E \\ 0 \\ 0 \end{bmatrix}$$

No zero diagonal elements will be encountered during the course of Gaussian elimination.

Example 2.3:

If our approach is used, initially, the matrix formulated is:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & \frac{-1}{R} & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -V_m \\ 1 & 0 & 0 & G_4 & 0 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & G_3 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & G_1 & 1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & -R_1 \end{bmatrix} \begin{bmatrix} I_E \\ I_1 \\ I_3 \\ V_4 \\ V_2 \\ V_3 \\ V_1 \\ I_1 \end{bmatrix} = \begin{bmatrix} E \\ E_1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

After interchanging rows, the resulting matrix is:

$$\begin{bmatrix} 1 & 0 & 0 & G_2 & 0 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & G_3 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & \frac{-1}{h} & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -r_m \\ 0 & -1 & 0 & 0 & 0 & 0 & G_1 & 1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & -R_2 \end{bmatrix} \begin{bmatrix} I_2 \\ I_1 \\ I_3 \\ V_4 \\ V_2 \\ V_3 \\ V_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ E \\ E_1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

No zero diagonal elements will be encountered during the course of Gaussian elimination.

These examples show that our approach indeed can avoid zero diagonal elements before the elimination and during the elimination. However, as mentioned before, if there are controlled sources or negative valued elements with specific set of element values, zero diagonal elements may be produced due to perfect cancellation.

2.4. Results

The implementation of this new algorithm into the DEC-10 version of SPICE2 has resulted in YSPICE. In YSPICE, the 'positive' nodes are first determined by the algorithm presented in Section 2.2. The network matrix is constructed using the element stamps as in [31]. The sparse matrix reordering is carried out using the Markowitz criterion [29,32] with diagonal pivoting. The row interchange is done by one extra set of pointers.

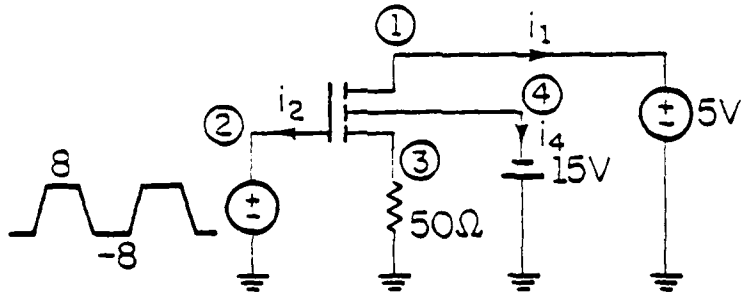
Examples which caused computational problems in the original version of SPICE2 due to pivoting on zero diagonal elements were successfully analyzed using YSPICE. Furthermore, the results we obtained show that in many cases the number of fills produced by our ordering strategy is far lower than that produced by previous methods, resulting in less computational cost, and at the same time, more accurate solutions.

Here a small selection of the examples analyzed by YSPICE is presented and the results are compared with those obtained by SPICE2.

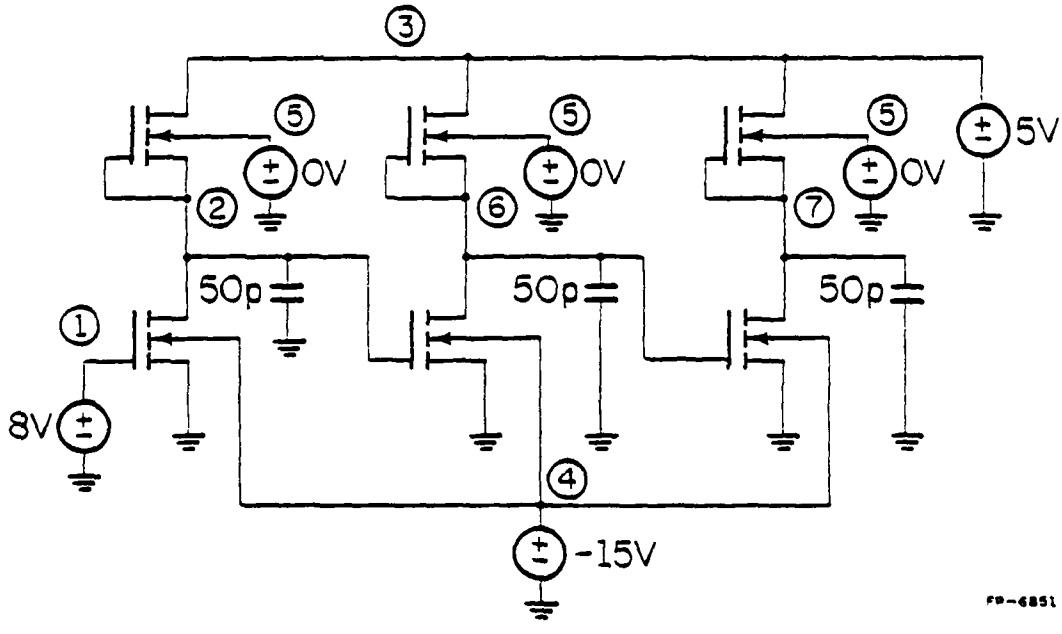
Example 2.4: The two circuits shown in Figs. 2.4(a) and (b) were analyzed using SPICE2 and YSPICE. The CPU times required by the equation solving subroutines in both programs for both circuits are given in Table 2.1.

Table 2.1 Simulation Data.

Circuit		CPU time for the equation solving subroutine	number of variables	number of operations per iteration
2.4(a)	YSPICE	0.9090 sec.	7	16
	SPICE2	1.9740 sec.	7	71
2.4(b)	YSPICE	0.031 sec.	10	30
	SPICE2	0.108 sec.	10	101



(a)



(b)

Fig. 2.4 Example Circuits.

The difference in the number of operations between YSPICE and SPICE2 in Table 2.1 can be explained as follows: In SPICE2, the matrix formulated by the modified nodal approach for the circuit in Fig. 2.4(a) is as shown in Fig 2.5(a). It can be seen that although the number of off-diagonal elements of the rows and columns corresponding to I_1, I_2, and I_4 is small, they are not chosen as pivots until their corresponding zero diagonal entries are filled. The delay causes the number of fills to increase greatly. In YSPICE, the matrix formulated for the circuit in Fig. 2.4(a) is as shown in Fig. 2.5(b). It can be seen that the number of fills is now zero due to the off-diagonal pivoting, and consequently, the number of operations is reduced.

Example 2.5: The circuit shown in Fig. 2.6 was also analyzed using both SPICE2 and YSPICE. The results of the dc analysis are shown in Table 2.2.

Table 2.2 Simulation Data.

Node	2	3	4	5	6
YSPICE node voltages	2.000 V	4.000 V	4.000 V	0.000 V	0.000 V
SPICE2 node voltages	2.000 V	2.324 V	2.324 V	-1.676 V	1675.9999 V

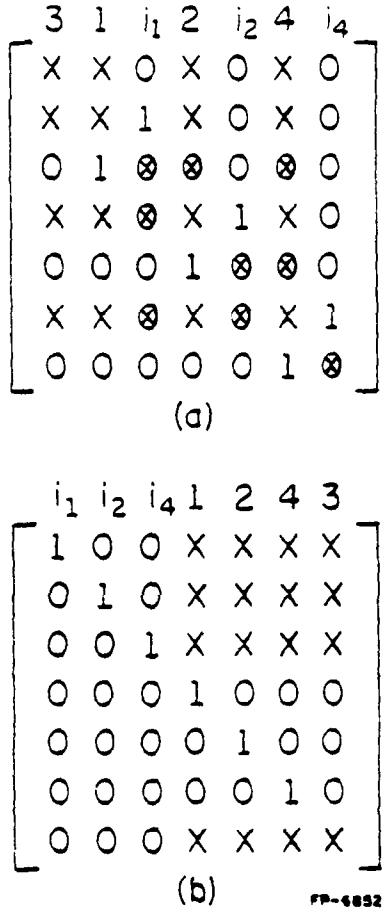


Fig. 2.5(a) Structure of the Network Matrix for the Circuit in Fig. 2.4(a) formulated by SPICE2.

(b) Structure of the Network Matrix for the Circuit in Fig. 2.4(a) formulated by YSPICE.

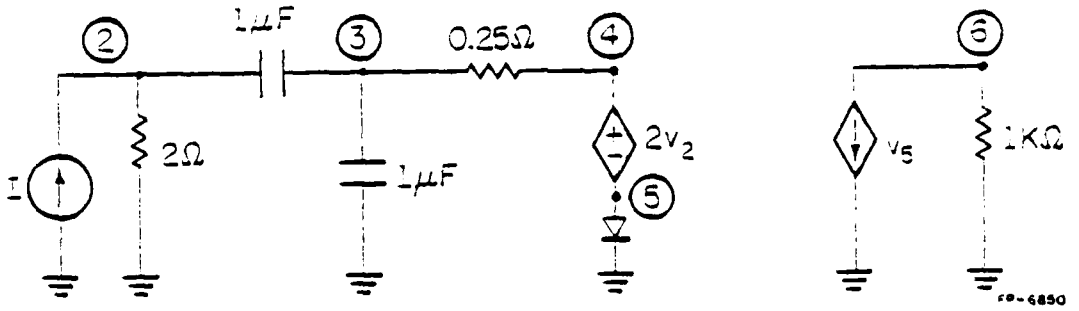


Fig. 2.6 Circuit used in Example 2.5.

Our approach gave correct results for this circuit while SPICE2 gave inaccurate results. These inaccuracies can be explained as follows: In SPICE2, if a diagonal element becomes too small, then it is replaced by 1.0×10^{-12}. In this circuit this approach is equivalent to connecting a $1.0\times10^{12}\Omega$ resistor from node 4 to ground. In this circuit the diode is reverse biased, the equivalent resistance used in SPICE2 for this diode is $0.721\times10^{12}\Omega$, as a result the computed I_1 in SPICE2 is 2.324×10^{-12}A, instead of the correct value, which should be 0.0A. This inaccuracy in computing I_1 makes $V_5 = -1.6760$V and $V_6 = 1675.9999$V instead of 0.0V.

2.5 Discussion

In this chapter we have presented an ordering stategy to be followed when the modified nodal approach is used. When this new strategy is used, the possibility of selecting zero diagonal pivots is reduced. The new strategy eliminates the need for having to continuously check the pivot and to replace it by a nonzero value in case a zero is generated, as is done in some existing strategies, which is both time consuming and inaccurate.

In addition, if the currents through the voltage sources are not needed, our ordering scheme provides a convenient way of reducing computation by performing the backward substitution step only partially to obtain the required variables.

Although by performing off-diagonal pivoting, the circuit matrix loses its symmetry and increases the complexity of the program, however, this is not a serious drawback. In fact, in many of the examples which we have analyzed, we have observed that by using off-diagonal pivoting, the number of fills is much less than that produced by other methods.

III. MODIFIED NEWTON METHOD AND PIECEWISE-NONLINEAR APPROACH

In a computer-aided circuit simulation program, if a circuit contains nonlinear elements, then a nonlinear solution method is required to solve the nonlinear algebraic equations in both dc analysis and transient analysis of the circuit. There are many nonlinear solution methods available, but the one most widely used is the Newton-Raphson method. This method has the desirable property that its rate of convergence is quadratic in the neighborhood of the solution.

Although The Newton-Raphson method has excellent local convergence properties, it has problems [1,33] when the initial guess is not close to the solution, such as numerical overflow, slow convergence, or no convergence. Several modified Newton-Raphson methods have been proposed to try to resolve the above problems, and the performance of the basic Newton-Raphson method has been improved to some extent. Here a new method - the piecewise nonlinear approach - is presented, and examples are given which show even further improvement. This method evolved from the piecewise linear method and previous modified Newton-Raphson methods, so it has the advantages of both methods. However, this new method is still at the experimental stage, no definite conclusion about it has been obtained.

This chapter begins with the introduction of the Newton-Raphson method. In Section 3.2, problems with the Newton-Raphson method are illustrated. In Section 3.3 the piecewise nonlinear approach is presented. In Section 3.4, a new modified Newton-Raphson method for bipolar devices is detailed and the piecewise nonlinear approach for bipolar devices including the avalanche effect is given in Section 3.5. In Section 3.6, the

piecewise nonlinear approach for the MOSFET is described. In Section 3.7, a discussion of the piecewise nonlinear approach is given.

3.1. The Newton-Raphson Algorithm

Let the set of nonlinear equations be

$$\underset{\sim}{F}(\underset{\sim}{X}) = \underset{\sim}{0} \tag{3.1}$$

If $\underset{\sim}{X}_k$ is the solution at the kth iteration, from Taylor series expansion, we have

$$\underset{\sim}{F}(\underset{\sim}{X}) = \underset{\sim}{F}(\underset{\sim}{X}_k) + \underset{\sim}{J}(\underset{\sim}{X}_k)\,(\underset{\sim}{X} - \underset{\sim}{X}_k) + \text{higher order terms} \tag{3.2}$$

Eq. (3.2) is used to obtain a solution to Eq. (3.1) under the assumption that the higher order terms are negligible. Thus, we write

$$\underset{\sim}{F}(\underset{\sim}{X}_k) + \underset{\sim}{J}(\underset{\sim}{X}_k)\,(\underset{\sim}{X}_{k+1} - \underset{\sim}{X}_k) = \underset{\sim}{0} \tag{3.3}$$

Solving Eq. (3.3) for $\underset{\sim}{X}_{k+1}$ we obtain

$$\underset{\sim}{X}_{k+1} = \underset{\sim}{X}_k - [\underset{\sim}{J}(\underset{\sim}{X}_k)]^{-1}\underset{\sim}{F}(\underset{\sim}{X}_k) \tag{3.4}$$

Eq. (3.4) is called the Newton-Raphson iteration algorithm.

3.2. Problems with the Newton-Raphson Algorithm

The problems of numerical overflow and slow convergence can be illustrated by a simple diode circuit shown in Fig. 3.1. The branch constraint for the typical semiconductor diode has the form $i = I_s(e^{40v} - 1)$ Given an initial estimate V_o to the solution for this circuit as shown in Fig. 3.1, it is not uncommon for the solution V_1 to the next

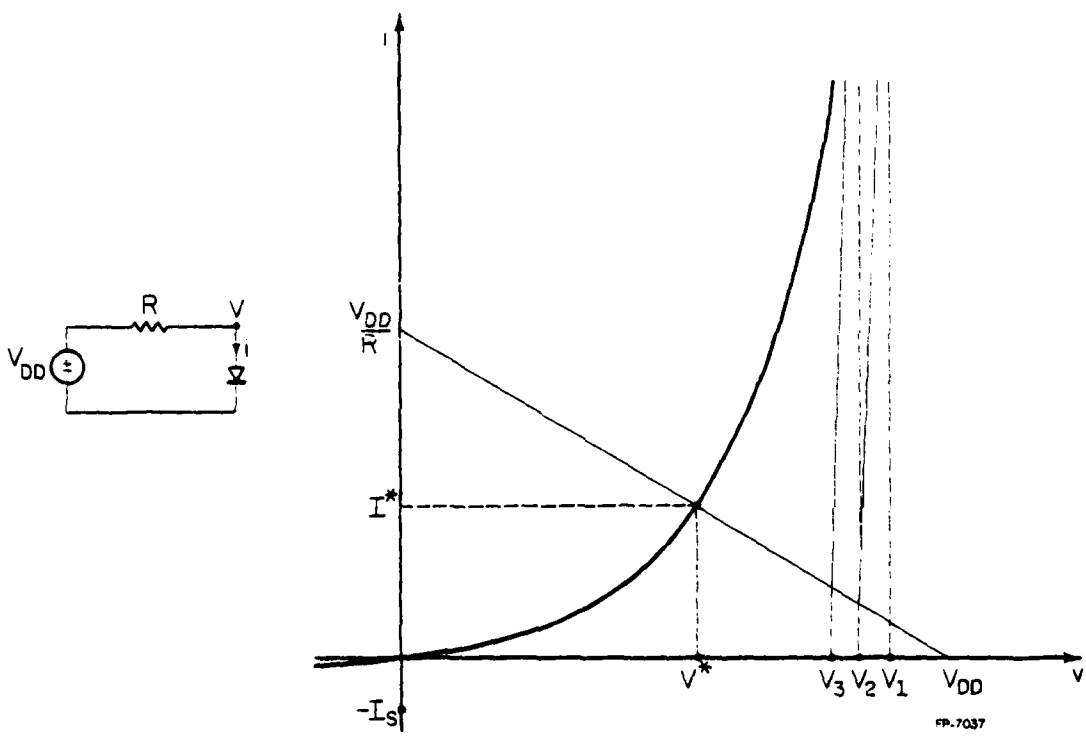


Fig. 3.1 Overflow and Slow Convergence Problem with the Simple Diode Circuit.

Newton-Raphson iterate to be in the neighborhood of V_{DD} as shown in Fig. 3.1. If the exponent in the diode equation is too large, overflow may occur. Even if overflow does not occur, convergence will be extremely slow because of the very large slope of the diode characteristic in this region. One modified Newton-Raphson algorithm which has proved successful in avoiding the above problems was proposed by Colon [33]. In this algorithm iteration on current is employed if V_{k+1} exceeds a reference junction voltage V_{REF}, this is illustrated in Fig. 3.2. This algorithm is used in the SPICE2 program.

Another problem with the Newton-Raphson algorithm is the lack of convergence. This is illustrated in Fig. 3.3. The iterate solutions will oscillate between V_o and V_1 and never converge to the solution V^*.

3.3. Piecewise Nonlinear Approach

This is a new approach which has the advantages of the piecewise linear approach and the modified Newton-Raphson methods. However, this method is still at the experimental stage, the proof of global convergence or conditions for global convergence has not been obtained. We restrict our discussion to two terminal elements. In this approach, first, a set of breakpoints is chosen and the device characteristic is partitioned into several nonlinear pieces. The partition must satisfy the following constraints:

(1) each piece must be monotonic and the first derivative must be monotonic too;

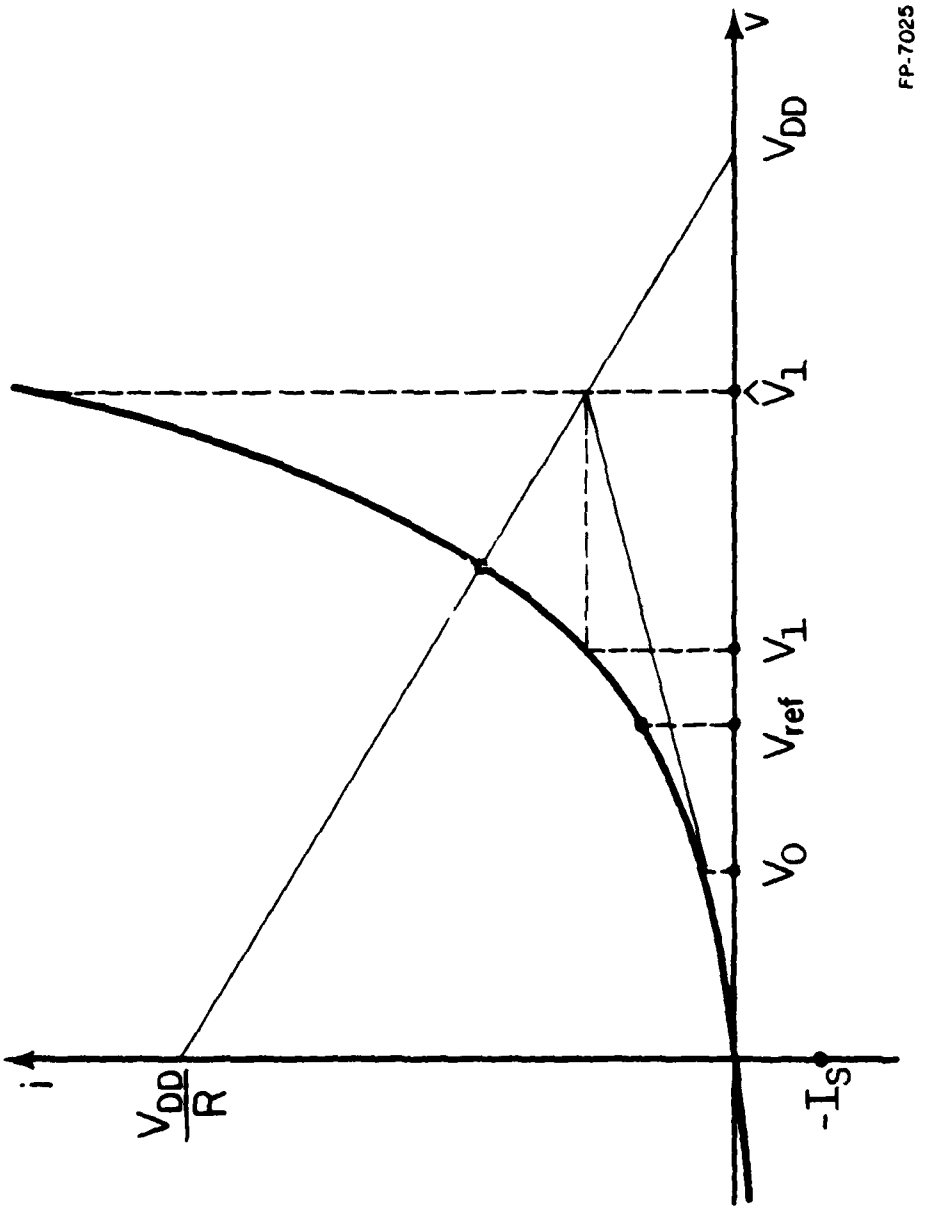


Fig. 3.2 Comparison of Current Iteration and Voltage Iteration.

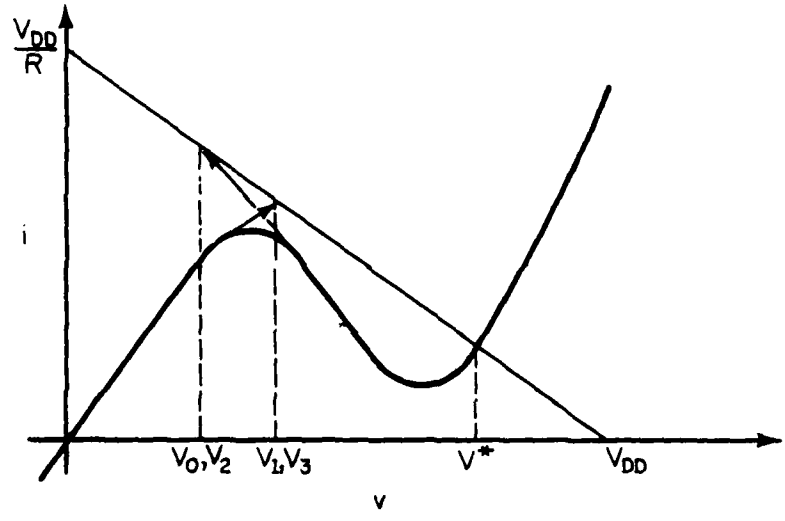


Fig. 3.3 Example of a Tunnel Diode Circuit.

(2) the piece must be chosen to be suitable for the current/voltage iteration to avoid numerical overflow and to hasten convergence;

(3) the number of pieces must be kept as small as possible to avoid the possibility of slow convergence.

After the partitioning, the following algorithm is used to perform the iteration:

(1) choose an initial guess V_o;

(2) linearize the circuit by Newton-Raphson method and find the iterate solution V_{k+1} ($k = 0$);

(3) if V_{k+1} is within the original piece, then use the modified Newton-Raphson method to choose V_{k+1}, and continue the iteration; otherwise, if the next breakpoint in the direction of change has not been chosen before, choose V_{k+1} equal to it; otherwise go into the adjacent piece, if the derivative is not continuous at this breakpoint, then choose this breakpoint as V_{k+1} again but use the new derivative; otherwise, choose the other breakpoint as V_{k+1} and continue the iteration.

This approach is illustrated in the graphical solution that is given in Fig. 3.4. Here the tunnel diode characteristic is partitioned into four pieces. The initial guess is V_o located in piece I. The solution of the linearized circuit is $\hat{V}_1$ which is not in piece I, so V_1 is chosen to be equal to breakpoint 1. The solution of the new linearized circuit is $\hat{V}_2$ which is still not in piece I. Enter piece II and choose breakpoint 2 as V_2. The iterate solution $\hat{V}_3$ is not in piece II, go into piece III and choose breakpoint 3 as V_3, the iterate solution $\hat{V}_4$ is not in piece III.

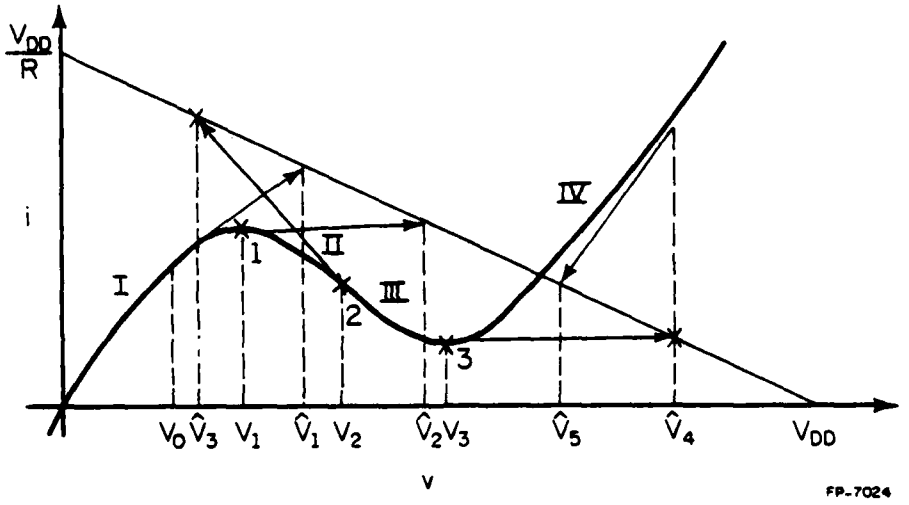


Fig. 3.4 Example of the Piecewise Nonlinear Approach.

Enter piece IV and choose $\hat{V}_4$ as V_4, this time, the solution $\hat{V}_5$ is in piece IV. Continue the iteration by modified Newton-Raphson method until the convergence is obtained.

3.4. A New Modified Newton-Raphson Method for Bipolar Devices

In the piecewise nonlinear approach, the diode characteristic is partitioned into three pieces as shown in Fig. 3.5. In region III, in order to avoid numerical overflow and to compensate for large higher order terms, the modified Newton-Raphson method must be used [1,33,34].

Consider the simple diode circuit shown in Fig. 3.6. The nodal equation is

$$F(V) = \frac{V - V_{DD}}{R} + I_s(e^{V/V_t} - 1) = 0 \tag{3.5}$$

By a Taylor series expansion we obtain

$$F(V_{k+1}) = F(V_k) + F'(V_k)\,(V_{k+1} - V_k) + \frac{F''(V_k)}{2}(V_{k+1} - V_k)^2 + \text{higher order terms} \tag{3.6}$$

where $F'(V_k)\,(V_{k+1} - V_k) = \left(\frac{1}{R} + \frac{I_s}{V_t} e^{V_k/V_t}\right)(V_{k+1} - V_k)$ and

$$\frac{F''(V_k)}{2}(V_{k+1} - V_k)^2 = \frac{I_s}{2*V_t^2} e^{V_k/V_t}\,(V_{k+1} - V_k)^2.$$

If we assume that R is sufficiently large, then the ratio of the third term to the second term in Eq. (3.6) is

$$\frac{(V_{k+1} - V_k)}{2*V_t} \tag{3.7}$$

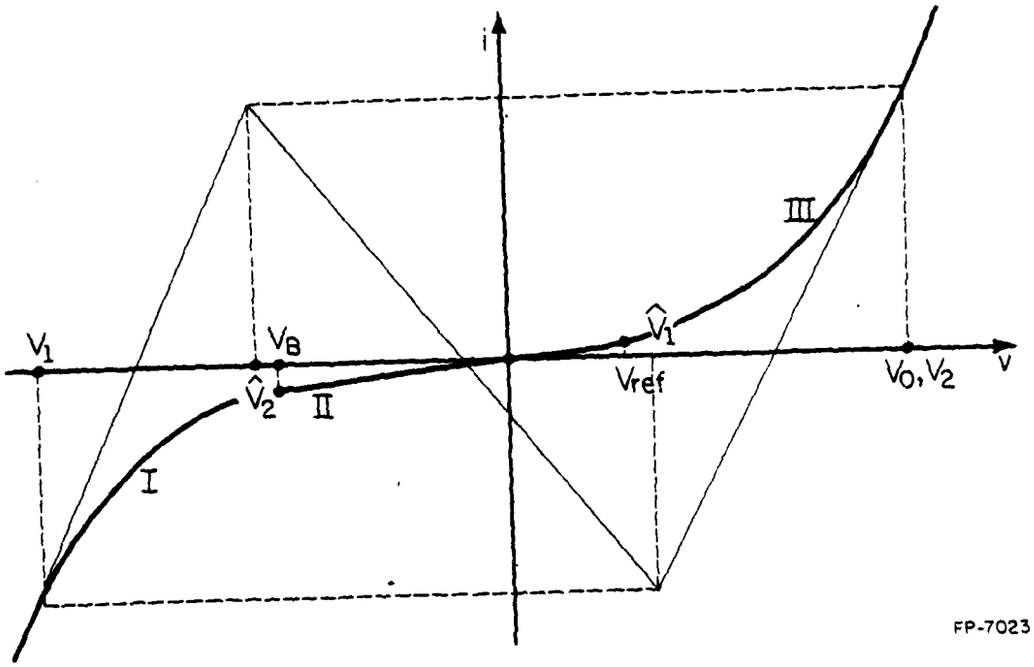


Fig. 3.5 Diode Static I-V Characteristic.

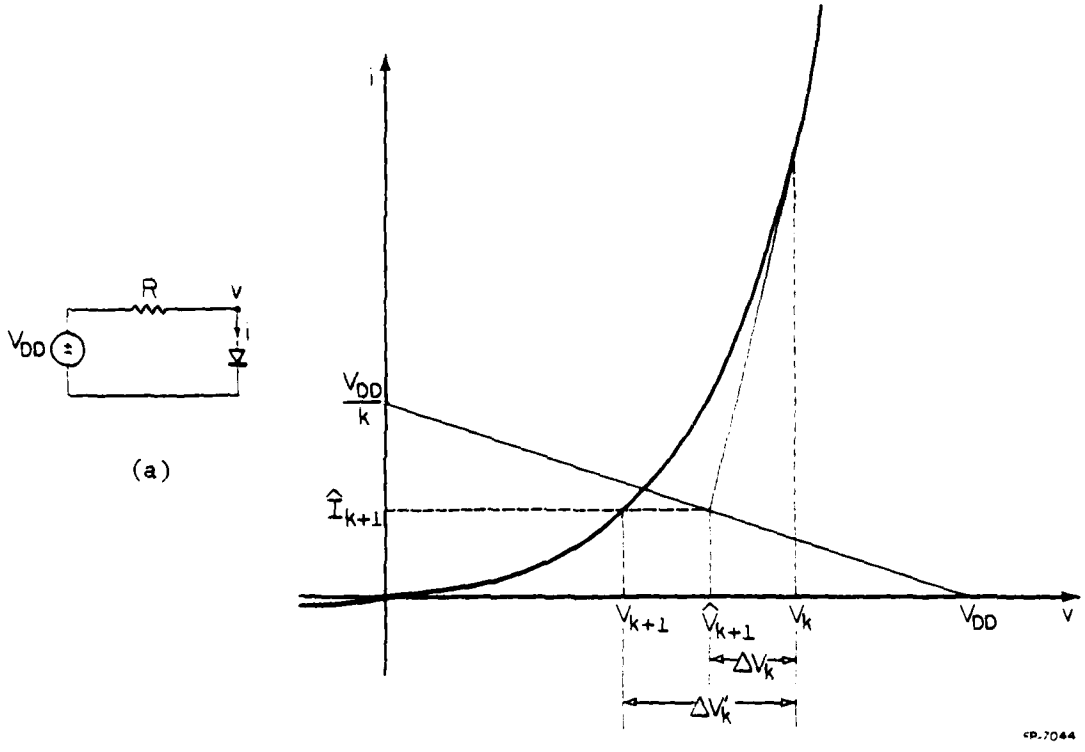


Fig. 3.6(a) Simple Diode Circuit.

(b) Newton-Raphson Iteration Solutions for (a).

If $(V_{k+1} - V_k)$ is not small compared to $2V_t$, then the assumption that the higher order terms in Eg. (3.2) are small and can be neglected is not true, so the correction term $\Delta V_k = V_{k+1} - V_k$ obtained by the Newton-Raphson method may not be good.

Let $\Delta V_k'$ be the modified correction term such that

$$F(V_k + \Delta V_k') = \frac{V_k+\Delta V_k'-V_{DD}}{R} + I_S(e^{(V_k+\Delta V_k')/V_t}-1) = 0 \tag{3.8}$$

Because ΔV_k satisfies

$$F(V_k) + F'(V_k)\Delta V_k=0 \tag{3.9}$$

so

$$F(V_k) + F'(V_k)\Delta V_k = F(V_k + \Delta V_k') \tag{3.10}$$

From Eq. (3.10) we obtain

$$\frac{(\Delta V_k'-\Delta V_k)}{R} + I_s e^{(V_{k+}\Delta V_k')/V_t} = I_S e^{V_k/V_t}(1+ \frac{\Delta V_k}{V_t}) \tag{3.11}$$

From Eq. (3.11) we obtain

$$1+ \frac{\Delta V_k}{V_t} - e^{\Delta V_k'/V_t} = \frac{\frac{\Delta V_k'-\Delta V_k}{R}}{I_S e^{V_k/V_t}} \tag{3.12}$$

If $(\Delta V_k' - \Delta V_k)/R$ is sufficiently small compared to the exponential term $I_s e^{V_k/V_t}$, then the equation

$$\Delta V_k' = V_t \ln(1+ \frac{\Delta V_k}{V_t}) \tag{3.13}$$

yields a good approximation to the true solution.

Now we would like to find out the relationship of Eq. (3.13) to current iteration. For current iteration, after obtaining

$$\hat{V}_{k+1} = V_k + \Delta V_k \tag{3.14}$$

we can obtain

$$\hat{I}_{k+1} = I_s(e^{V_k/V_t} - 1) + \frac{I_s}{V_t} e^{V_k/V_t} \Delta V_k \tag{3.15}$$

If $\hat{I}_{k+1} > -I_s$, then there is a point $V_{k+1} = V_k + \Delta\hat{V}_k$ on the diode charateristic whose current is $\hat{I}_{k+1}$.

$$I_s(e^{(V_k+\Delta\hat{V}_k)/V_t} - 1) = I_s(e^{V_k/V_t} - 1) + \frac{I_s}{V_t} e^{V_k/V_t} \Delta V_k \tag{3.16}$$

We obtain

$$\Delta\hat{V}_k = V_t \ln(1 + \frac{\Delta V_k}{V_t}) \tag{3.17}$$

We can see that Eq. (3.17) is identical to Eq. (3.13), also we can see that the condition for Eq. (3.16) to have a solution is

$$1 + \frac{\Delta V_k}{V_t} > 0 \quad . \tag{3.18}$$

Since if $\Delta V_k \geq 0$, then Eq. (3.18) is satisfied, so we only need to consider the situation when $\Delta V_k < 0$. This condition can be explained graphically in Figs. 3.7(a), (b) and (c). From Eq. (3.15) we see that $-I_s \leq \hat{I}_{k+1} \leq I_L$ for $-V_t \leq \Delta V_k \leq 0$. Thus, if the current intercept of the load line with the linearized diode curve lies in this range, then $|\Delta V_k|$ cannot exceed V_t and convergence can be quite slow if voltage iteration is

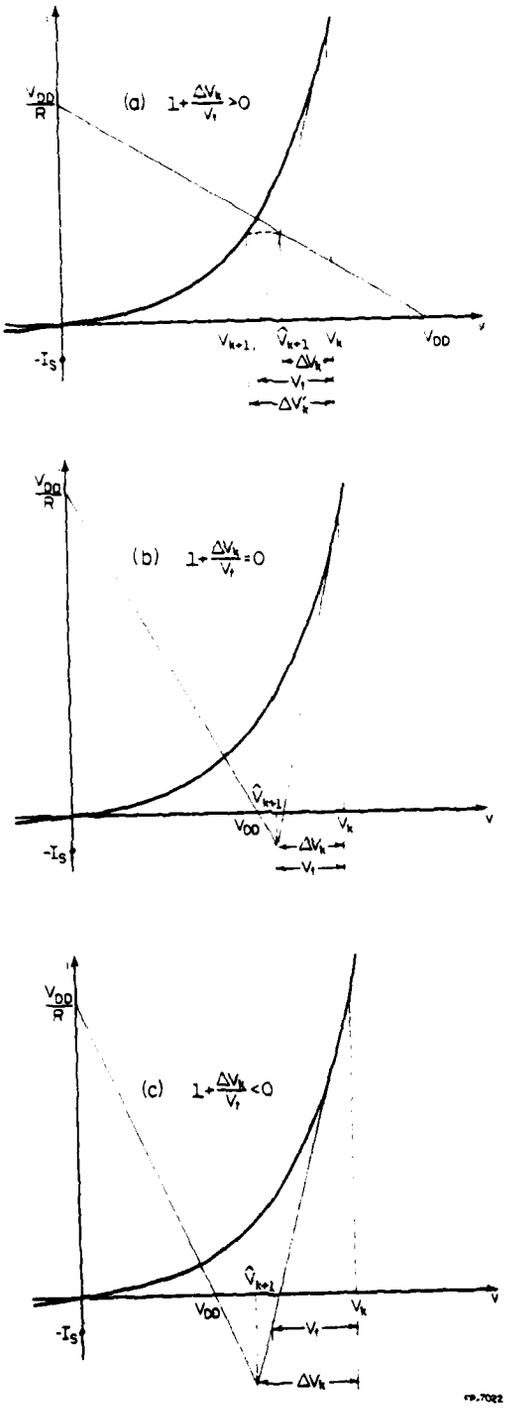


Fig. 3.7 Three Cases with the Simple Diode Circuit.

used.

For Example in Fig. 3.7(a) we see that $\hat{I}_{k+1} > -I_s$ and so

$$-V_t < \Delta V_k \leq 0$$

In Fig. 3.7(b) $\hat{I}_{k+1} = -I_s$ and so

$$\Delta V_k = -V_t$$

In Fig. 3.7(c) $\hat{I}_{k+1} < -I_s$ and so

$$\Delta V_k < -V_t$$

In cases (a) and (b) $|\Delta V_k| \leq V_t$, therefore # of iterations $\geq \left|\frac{V^* - V_k}{V_t}\right|$ if voltage iteration is used.

Let us consider the conditions for cases (a) and (b) to be true. From Eq. (3.9), we obtain

$$\frac{-\Delta V_k}{V_t} = \frac{(V_k - V_{DD})/R + I_s(e^{V_k/V_t} - 1)}{1/R + \frac{I_s}{V_t} e^{V_k/V_t}} \tag{3.19}$$

$$= \frac{1/R + \frac{I_s}{V_t} e^{V_k/V_t} + \frac{V_k - V_{DD} - V_t - R*I_s}{R*V_t}}{1/R + \frac{I_s}{V_t} e^{V_k/V_t}} \tag{3.20}$$

Since

$$\frac{V^* - V_{DD}}{R} + I_s(e^{V^*/V_t} - 1) = 0 \qquad (3.21)$$

so from Eqs. (3.19), (3.20) and (3.21), we obtain the following conclusions:

(1) If $V_k \geq V^*$ then $\Delta V_k \leq 0$.

(2) If $V_k \leq V^*$ and $R \geq \frac{V_k - V_{DD} - V_t}{I_s}$, then $-V_t \leq \Delta V_k \leq 0$. (3.22)

(3) If R and V_k are chosen such that $V_k \geq V^*$ and $R \geq \frac{V_k - V_{DD} - V_t}{I_s}$, and voltage iteration is used in the forward region, then the number of iteration is lowerbounded by $\left|\frac{V^* - V_k}{V_t}\right|$.

For example, if $V_k - V^* = 1.0V$ and $V_t = 0.025V$,

then $\left|\frac{V^* - V_k}{V_t}\right| = 40.$

The above conclusions show why current iteration must be used in the forward region.

Now we would like to examine under what conditions current iteration should be used and if there is a V_{REF} (such as the V_{REF} used in SPICE2) to determine whether current iteration or voltage iteration should be used.

Let us consider the simple diode circuit in Fig. 3.8. V_k, V^* and $\hat{V}_k$ satisfy Eq. (3.23)

$$I_s(e^{V^*/V_t} - e^{\hat{V}_k/V_t}) = \frac{V_k - V^*}{R} \qquad (3.23)$$

Let us consider the limiting case when $V^* - V_k \ll V_t$, then Eq. (3.23) can be rewritten as:

$$\frac{R^* I_s}{V_t} e^{V^*/V_t}(V^* - \hat{V}_k) = V_k - V^* \qquad (3.24)$$

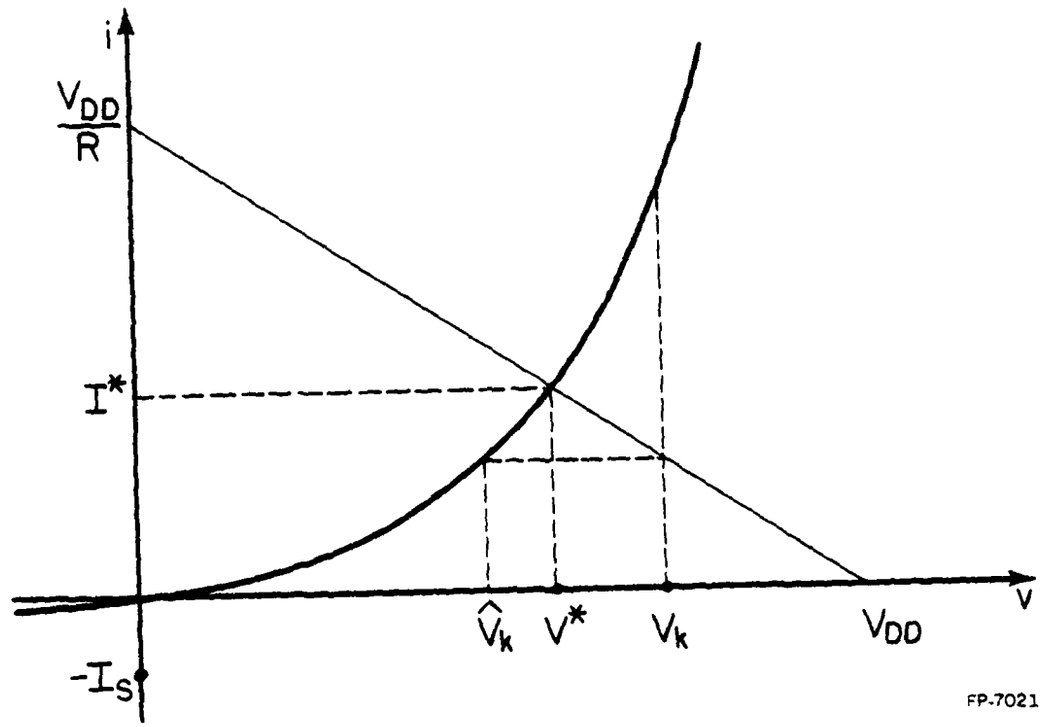


Fig. 3.8 Comparison of Current Iteration to Voltage Iteration.

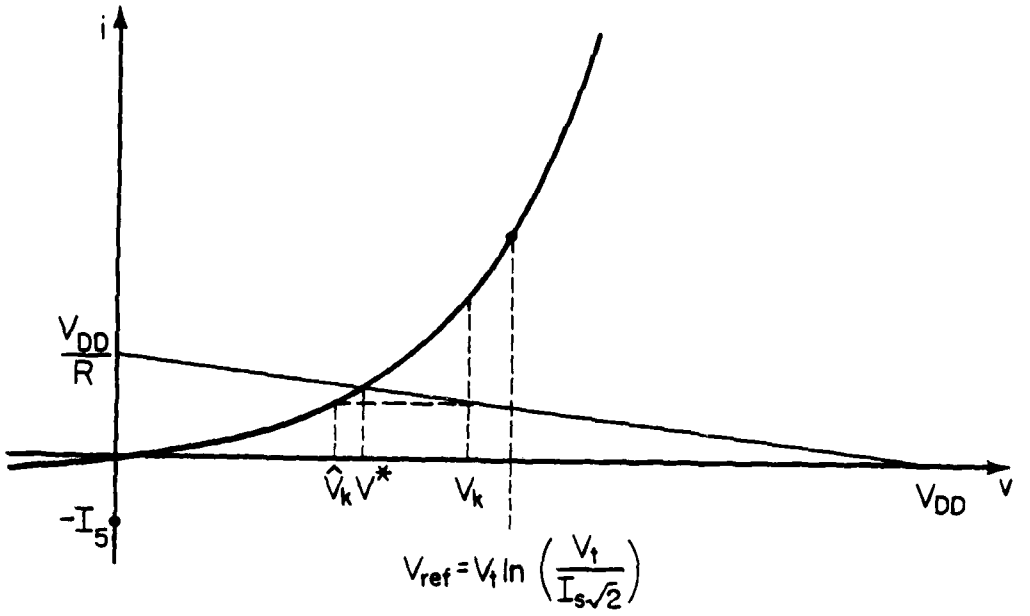
so if $\frac{R^*I_s}{V_t} e^{V^*/V_t} >> 1$, then current iteration is preferred; else if $\frac{R^*I_s}{V_t} e^{V^*/V_t} << 1$, then voltage iteration is preferred. These two conditions are illustrated in Figs. 3.9(a) and (b).

From Eq. (3.24) we can conclude that V_{REF} must satisfy

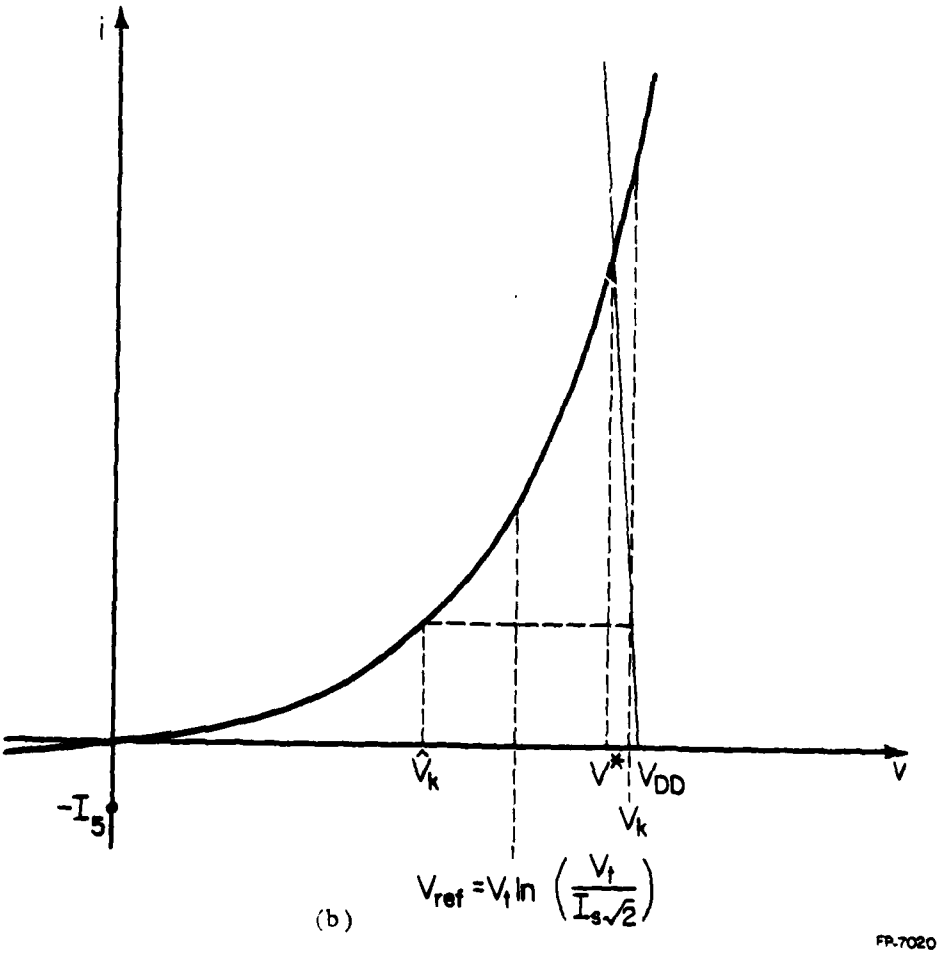
$$\frac{R^*I_s}{V_t} e^{V_{REF}/V_t} = 1 \tag{3.25}$$

Since the value of R is not a constant, there is no universal V_{REF}. Experiments of the simple diode circuit with different values of R and V_{DD} were done to test the above conclusion, and the data are given in Figs. 3.10(a), (b), (c), and (d). These data confirm Eq. (3.25). In Fig 3.10(a), $\frac{R^*I_s}{V_t} e^{V^*/V_t}$ is always much larger than one, this explains why current iteration is always better than voltage iteration; in Fig. 3.10(b), when V_{DD} is less than 0.7V, $\frac{R^*I_s}{V_t} e^{V^*/V_t}$ is less than one, voltage iteration is better than current iteration; in Fig. 3.10(c), when V_{DD} is less than 0.5V, $\frac{R^*I_s}{V_t} e^{V^*/V_t}$ is less than one, so voltage iteration is better than current iteration; in Fig. 3.10(d), because $\frac{\Delta V_k}{V_t}$ may be less than -1, strict current iteration in region III can not be done. Whenever $\frac{\Delta V_k}{V_t}$ is less than -1, the next guess is reset to zero, and current iteration is resumed. for this approach, current iteration is always better than voltage iteration.

In the conventional current/voltage iteration approach, such as the one used in SPICE2, there is a universal V_{REF}, if V_{k+1} exceeds V_{REF}, then current iteration is used; otherwise, voltage iteration is used. In SPICE2, this V_{REF} is set to the point of minimum radius of curvature:



(a)



(b)

FR-7020

Fig. 3.9(a) Situation When Current Iteration Is Better Than Voltage Iteration.
(b) Situation When Voltage Iteration Is Better Than Current Iteration.

(a)

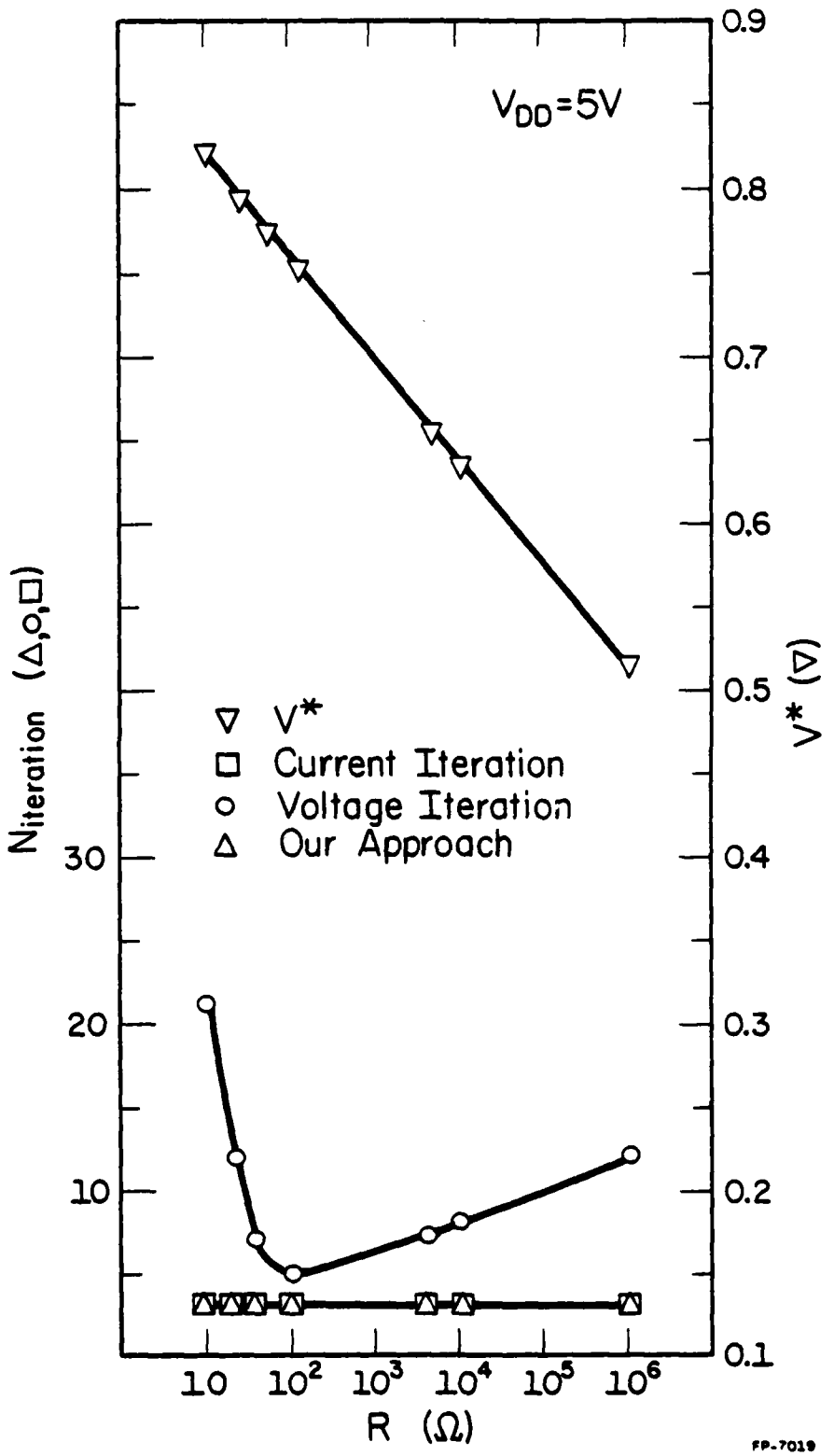


Fig. 3.10(a) Comparison of Iteration Methods for the Simple Diode Circuit.

(b)

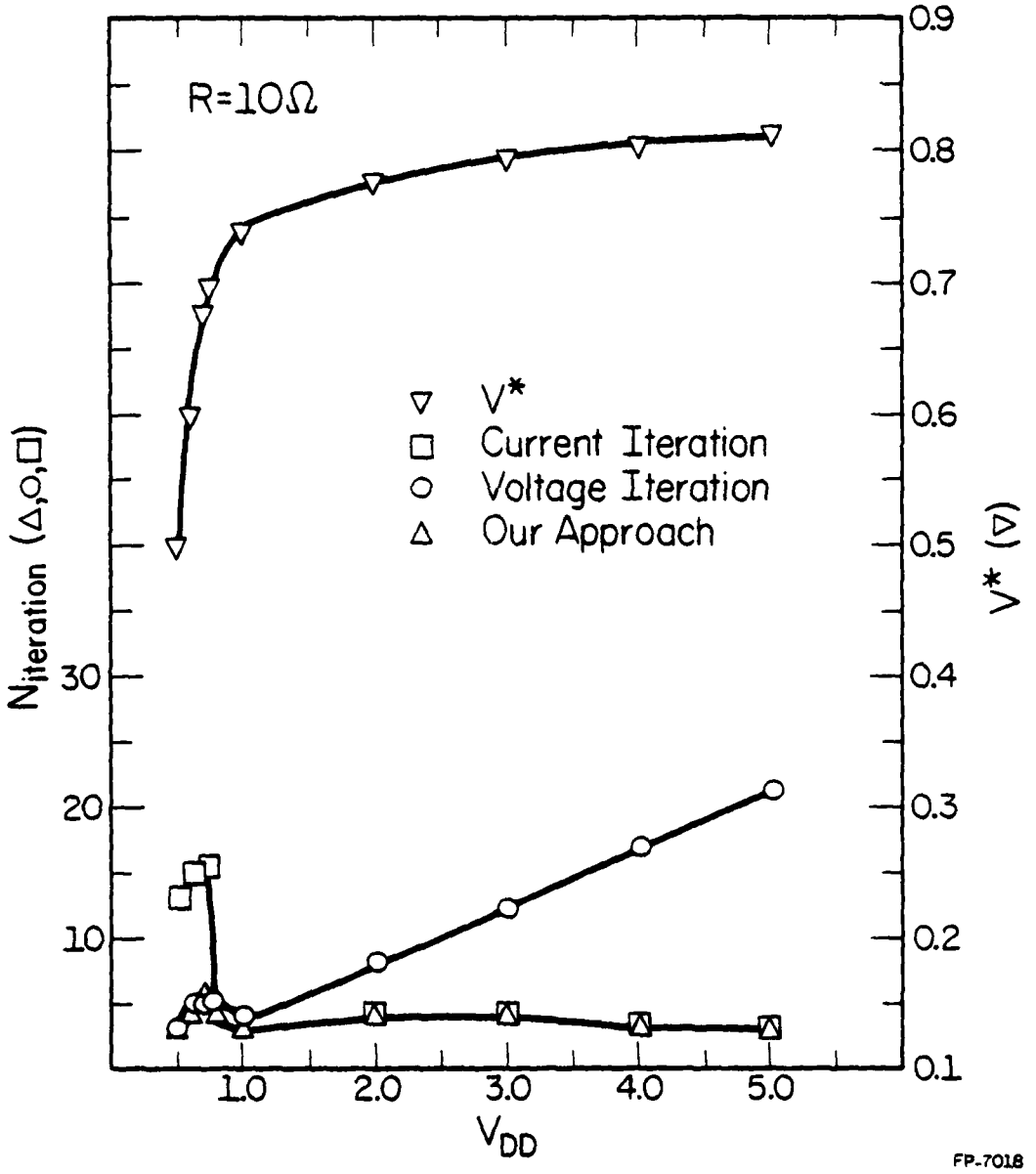


Fig. 3.10(b) Comparison of Iteration Methods for the Simple Diode Circuit.

(c)

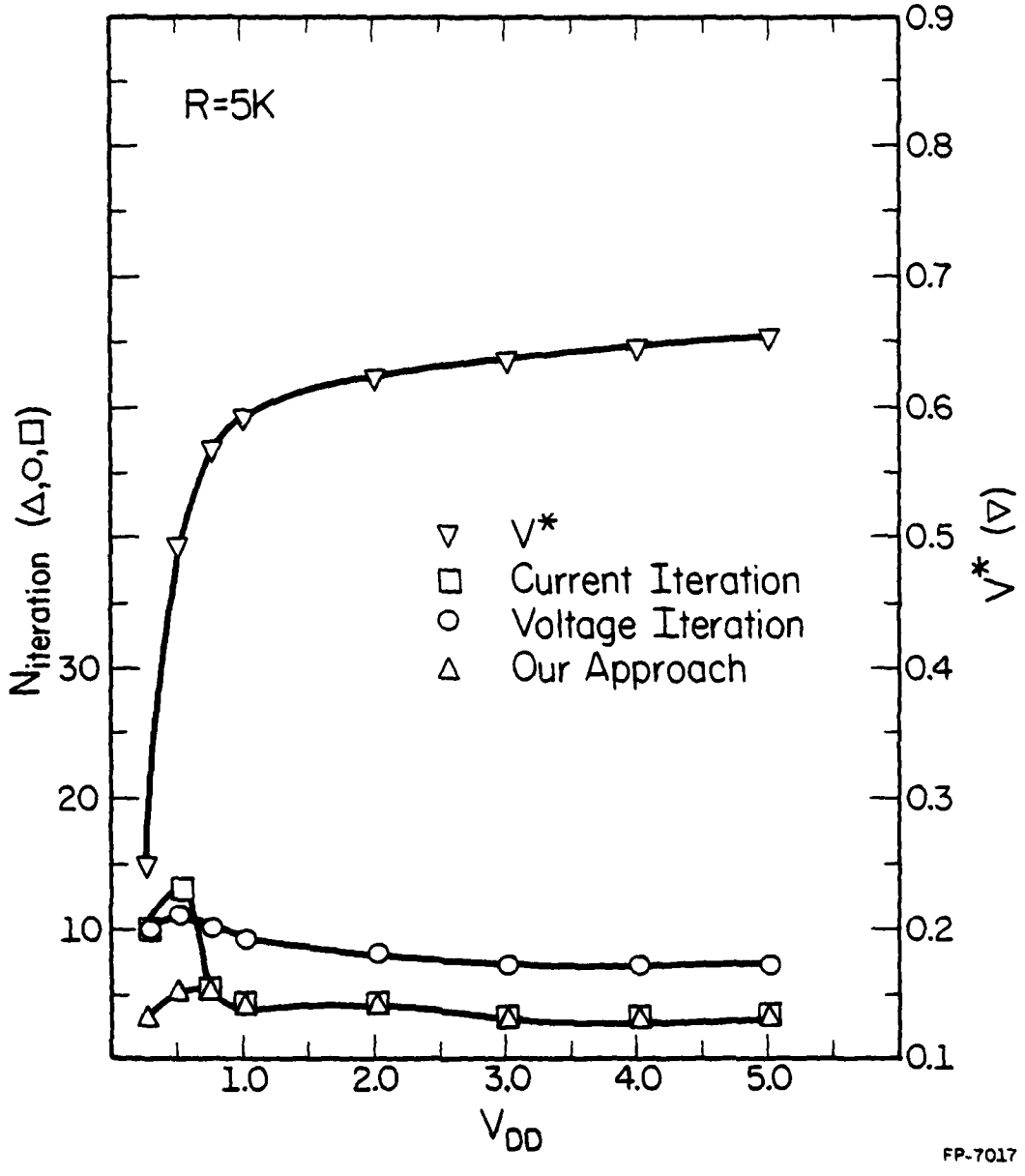


Fig. 3.10(c) Comparison of Iteration Methods for the Simple Diode Circuit.

(d)

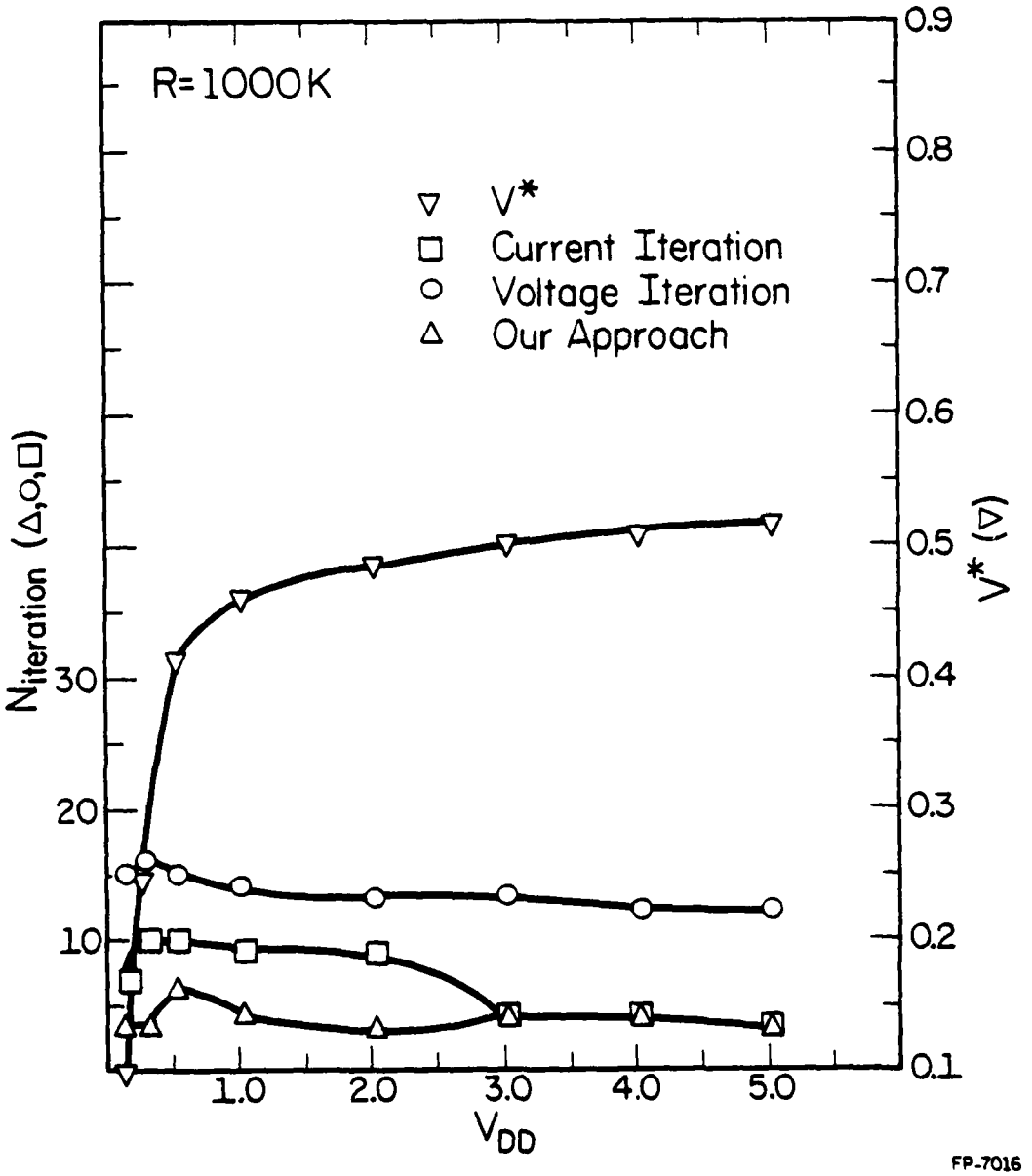


Fig. 3.10(d) Comparison of Iteration Methods for the Simple Diode Circuit.

$$V_{REF} = V_t \ln\left(\frac{V_t}{I_s\sqrt{2}}\right) \tag{3.26}$$

From the above analysis, we can see that this V_{REF} does not provide any guarantee of fast convergence. The simple diode circuit was used again to test the approach used in SPICE2, V_{DD} = 5V, R = 1000K, the number of iterations used by SPICE2 is 12, while the number of iterations for strict current iteration is only 3.

There is another problem associated with the conventional current/voltage iteration used in SPICE2. This problem is illustrated in Fig. 3.11. Let us assume that the initial guess is V_0 and the first iterate solution is V_1. Now we do not know which load lines we are encountering, because both load lines will give us V_1. If it is load line 1, then voltage iteration should be used; if it is load line 2, then voltage iteration is too slow. In SPICE2, because V_1 is less than V_{REF}, voltage iteration is used for both cases. Experimental results show that for V_{DD} = -5V and R = 1000K the number of iterations used by SPICE2 is 12. This problem can be solved by using the piecewise nonlinear approach. Whenever this situation occurs, then the next guess is changed to zero. If it is load line 1, the next iterate solution is in the first quadrant and voltage iteration is used to obtain the solution. If it is load line 2, the next iterate solution is in the third quadrant. The number of iterations used by recognizing that the load line 2 is being used and changing the next guess to zero is 3.

Also let us examine Eq. (3.13) again. When $\frac{\Delta V_k}{V_t}$ is positive and much smaller than 1, then $\Delta V_k' \approx \Delta V_k$. If the difference between $\Delta V_k'$ and ΔV_k is small compared to the iteration error tolerance, then there is no need to

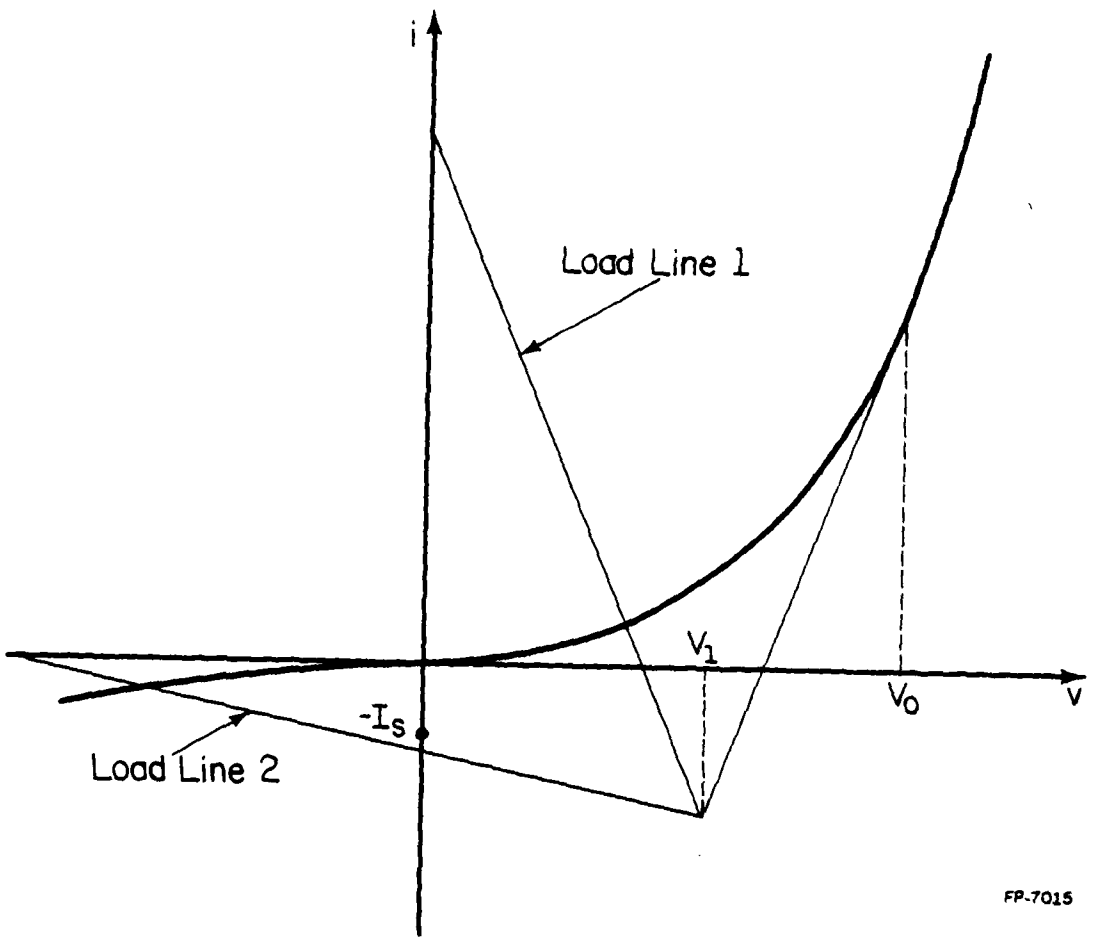


Fig. 3.11 Another Problem with the Colon Method.

do the transformation. Let us assume the error tolerance is 10^{-6}V, then when $\frac{\Delta V_k}{V_t}$ is less than 0.01, there is no need to do the transformation.

The result of all the above analysis is a new iteration scheme. The flowchart for this new approach is shown in Fig. 3.12(a), the experimental data for the simple diode circuit are also given in Figs. 3.10(a), (b), (c), and (d). These data show that this algorithm works well for resistor load diode circuits. However, when transistor circuits are solved, the load line generated by linearization changes during the iterations and the algorithm goes into limit cycle for some circuits. If the piecewise nonlinear method presented in Section 3.3 (it corresponds to a Katzenelson's type algorithm [40] for the piecewise linear approach) is used, then probably the limit cycle problem will not occur. But the piecewise nonlinear method only allows one diode to change regions at a given iteration, so the convergence rate is slow; also the piecewise nonlinear method requires a linear search to accomplish the task that only one diode changes regions. So instead of using a strict piecewise nonlinear approach, the algorithm in Fig. 3.12(a) was modified to eliminate the limit cycle problem. The flow chart for the modified algorithm is given in Fig 3.12(b).

Three test circuits were used to test this new iteration scheme. These three circuits are given in Fig. 3.13(a), (b) and (c), and the data are given in Table 3.1. These test results show that the new iteration scheme is superior to the Colon method used in SPICE2.

Fig. 3.12(a) Flowchart for the New Iteration Scheme.

(a)

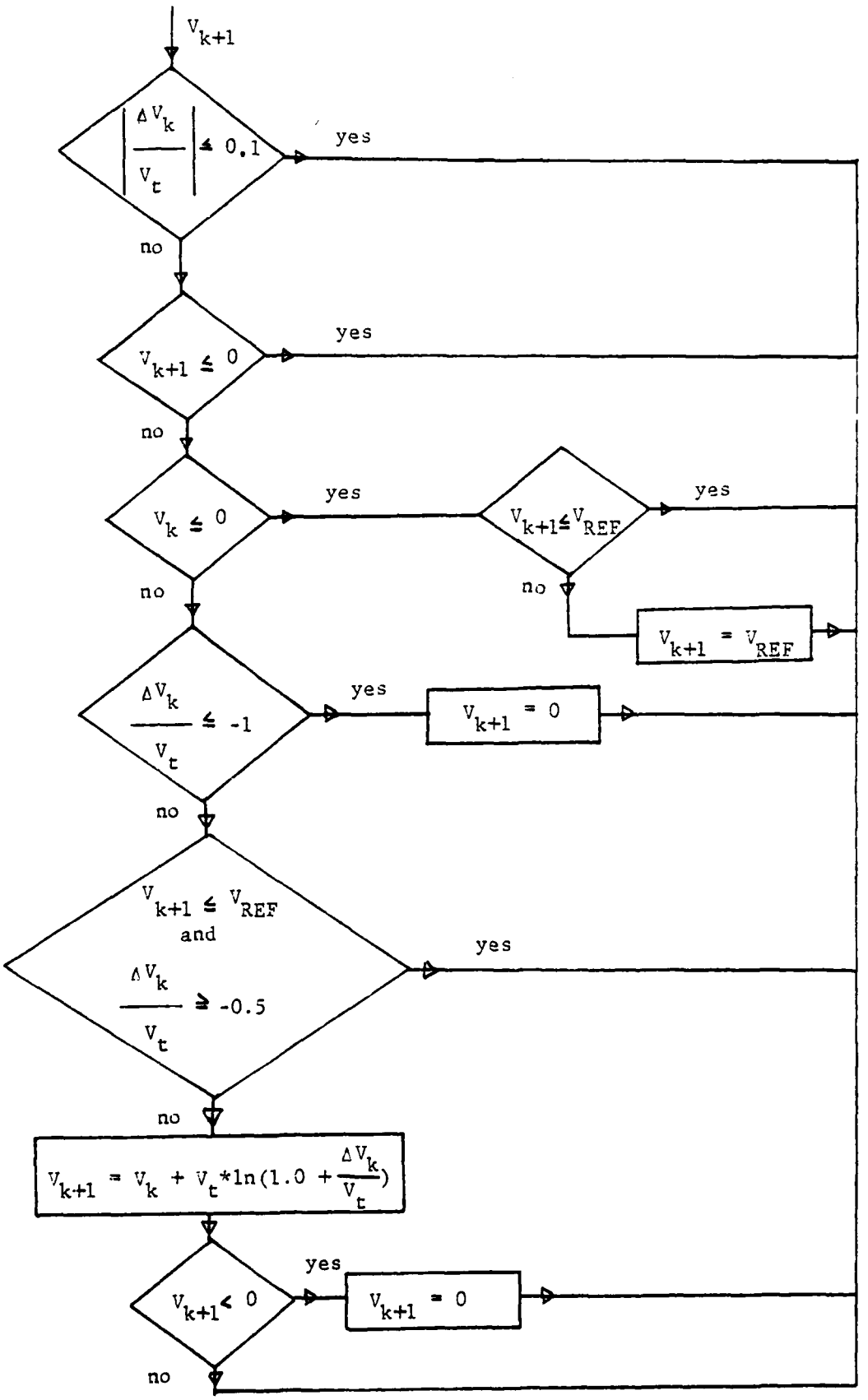

V_{k+1}
$\left|\frac{\Delta V_k}{V_t}\right| \leq 0.1$
yes
no
$V_{k+1} \leq 0$
yes
no
$V_k \leq 0$
yes
$V_{k+1} \leq V_{REF}$
yes
no
$V_{k+1} = V_{REF}$
$\frac{\Delta V_k}{V_t} \leq -1$
yes
$V_{k+1} = 0$
no
$V_{k+1} \leq V_{REF}$ and $\frac{\Delta V_k}{V_t} \geq -0.5$
yes
no
$V_{k+1} = V_k + V_t * \ln(1.0 + \frac{\Delta V_k}{V_t})$
$V_{k+1} < 0$
yes
$V_{k+1} = 0$
no

Fig. 3.12(b) Modified Flowchart for the New Iteration Scheme.

(b)

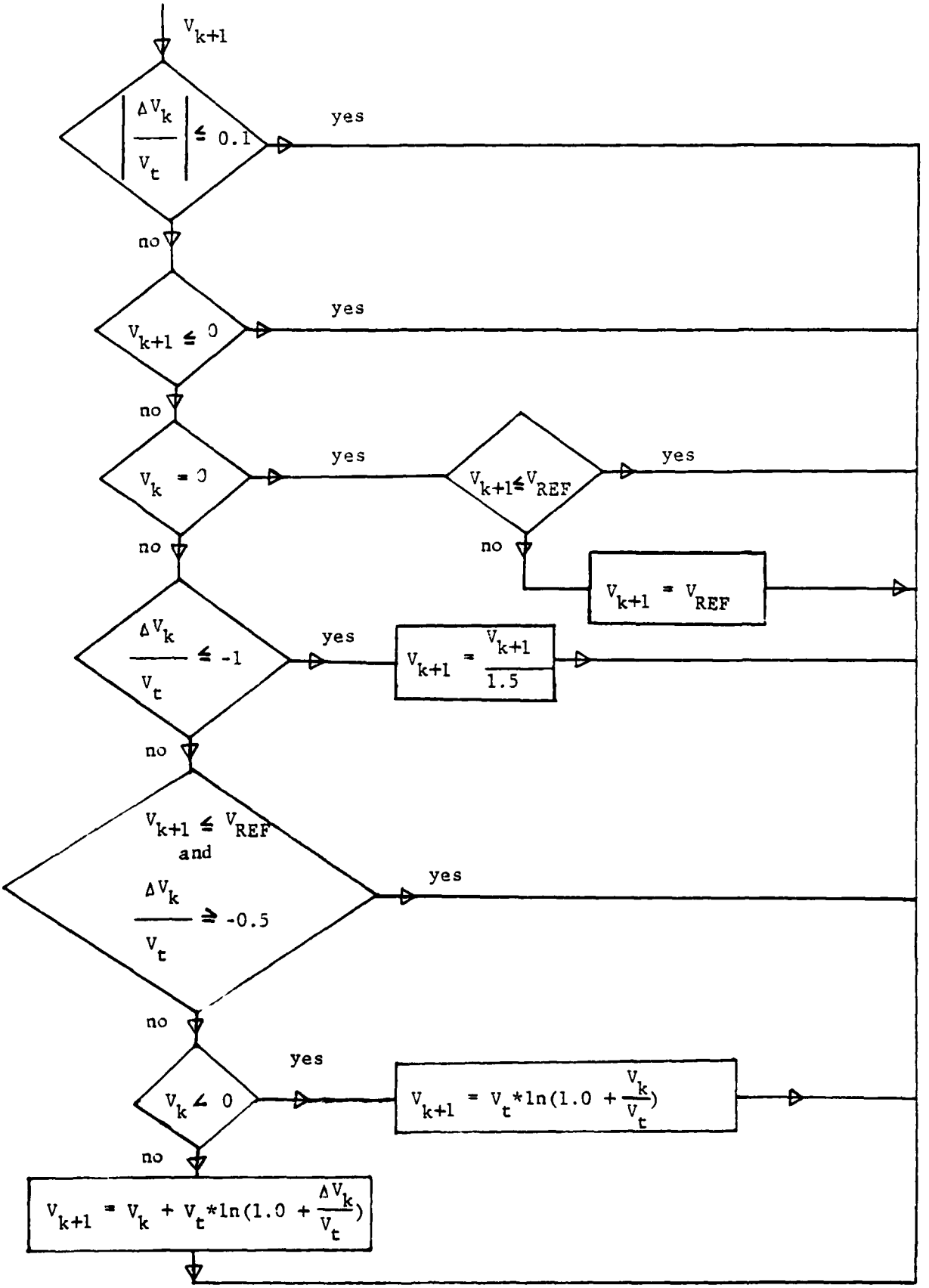
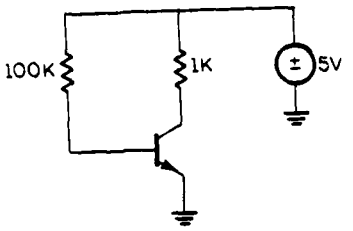
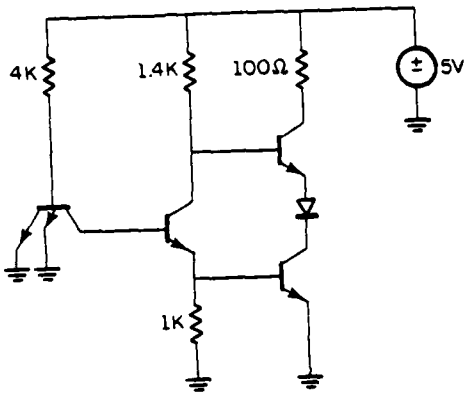

V_{k+1}
$\left|\frac{\Delta V_k}{V_t}\right| \leq 0.1$
yes
no
$V_{k+1} \leq 0$
yes
no
$V_k = 0$
yes
$V_{k+1} \leq V_{REF}$
yes
no
$V_{k+1} = V_{REF}$
$\frac{\Delta V_k}{V_t} \leq -1$
yes
$V_{k+1} = \frac{V_{k+1}}{1.5}$
no
$V_{k+1} \leq V_{REF}$ and $\frac{\Delta V_k}{V_t} \geq -0.5$
yes
no
$V_k < 0$
yes
$V_{k+1} = V_t * \ln(1.0 + \frac{V_k}{V_t})$
no
$V_{k+1} = V_k + V_t * \ln(1.0 + \frac{\Delta V_k}{V_t})$

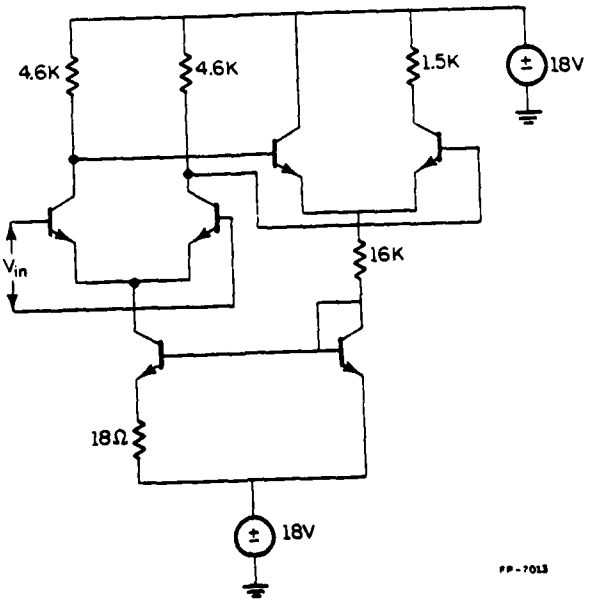
Fig. 3.13 Example Circuits (a) One Transistor Amplifier.
(b) TTL NAND Gate.
(c) Differential Amplifier.


100K
1K
5V

(a)


4K
1.4K
100Ω
5V
1K

(b)


4.6K
4.6K
1.5K
18V
16K
V_{in}
18Ω
18V
FP-7013

(c)

Table 3.1 Comparison of the Results between the New Approach and SPICE2.

Circuit	Number of iterations (New approach)	Number of iterations (SPICE2)
Fig. 3.13(a)	3	6
Fig. 3.13(b)	7	17
Fig. 3.13(c)	6	7

3.5. Piecewise Nonlinear Approach for Bipolar Devices Including Avalanche Effects

Although the avalanche charateristic of diode should consist of two separate exponential functions [35], in order to simplify the analysis, here the avalanche characteristic is chosen to consist of only one exponential function. The diode I-V static characteristic used here is shown in Fig. 3.5.

In the forward biased region the equation for the diode current is

$$I_d = I_s(e^{V_d/V_t} - 1) \tag{3.27}$$

The reverse-biased current before breakdown is

$$I_d = G_d V_d \tag{3.28}$$

The avalanche current is

$$I_d = -e^{A(V_B - B*V_d)} \tag{3.29}$$

The constants A and B are determined from the I-V characteristic curve, where V_B is the breakdown voltage and V_d is the junction voltage.

If, in order to hasten the convergence, strict current iteration is used in pieces I and III, then divergence may be encountered as shown in Fig. 3.5. Therefore, the piecewise nonlinear approach for bipolar devices with avalanche modeling is as follows:

(1) choose V_0 equal to V_{REF} and piece III;

(2) find iterate solution V_{k+1} by the new modified Newton method. If V_{k+1} is within piece III, repeat this step.

(3) otherwise go into piece II, choose $V_{k+1} = 0$ and use the new derivative, if V_{k+2} is within piece II, then solution is fo[illegible].

(4) otherwise, go into piece I, choose $V_{k+1} = V_B$, and use the new modified Newton method.

Remark: Only one nonlinear device is allowed to change its region at one iteration, otherwise, limit cycle problems may occur.

3.6. Piecewise Nonlinear Approach for MOSFET

Let us consider the simple resistor load MOS inverter which are shown in Fig. 3.14(a). The nodal equation is

$$F(V) = \frac{V - V_{DD}}{R} + \beta\left[(V_{IN} - V_T)V - \frac{V^2}{2}\right] \tag{3.30}$$

By Taylor series expansion we obtain

$$F(V_{k+1}) = F(V_k) + F'(V_k)\,(V_{k+1} - V_k) + \frac{F''(V_k)}{2}(V_{k+1} - V_k)^2 + \text{higher order terms} \tag{3.31}$$

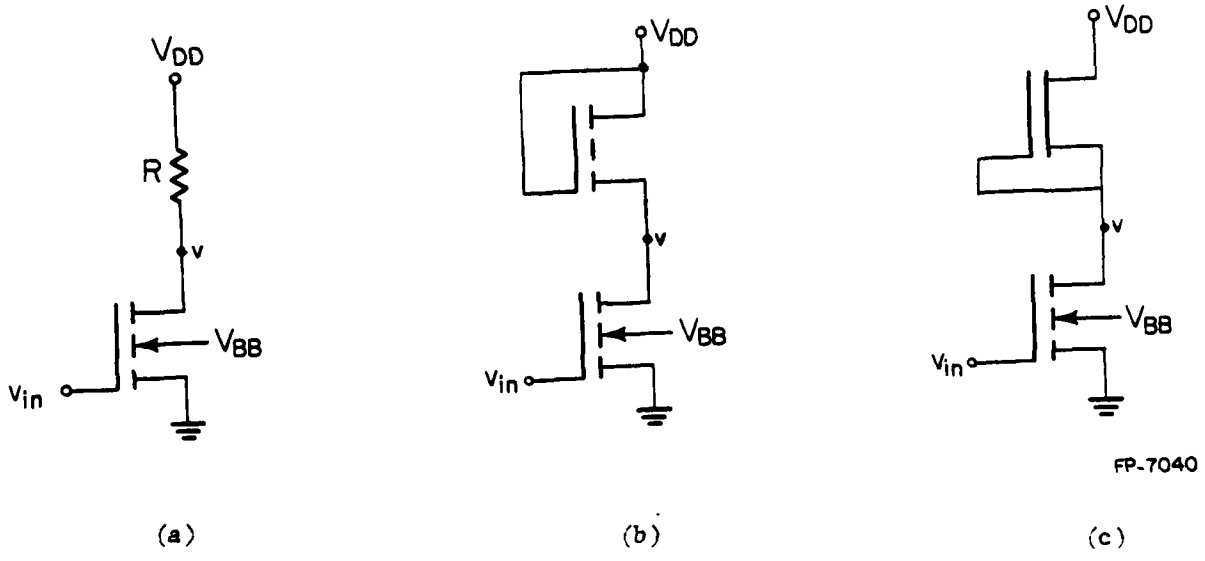
where $F'(V_k)\,(V_{k+1} - V_k) = \left[\frac{1}{R} + \beta(V_{IN} - V_T - V_k)\right](V_{k+1} - V_k)$ and

$$\frac{F''(V_k)}{2}(V_{k+1} - V_k)^2 = \frac{-\beta}{2}(V_{k+1} - V_k)^2.$$

If we assume R is sufficiently large, then

$$\frac{\text{the third term in Eq. (3.31)}}{\text{the second term in Eq. (3.31)}} = \frac{-(V_{k+1} - V_k)}{2(V_{IN} - V_T - V_k)} \tag{3.32}$$

If $(V_{k+1} - V_k)$ is not small compared to $2(V_{IN} - V_T - V_k)$, then the assumption that higher order terms in Eq. (3.12) are small and can be



(a) (b) (c)

Fig. 3.14(a) Resistor Load MOS Inverter Circuit.
(b) Saturated Load MOS Inverter Circuit.
(c) Depletion Load MOS Inverter Circuit.

neglected is not true, so the correction term $\Delta V_k = V_{k+1} - V_k$ obtained by the Newton-Raphson method is not good. This may result in very slow convergence. Fig 3.15(a) illustrates this problem. If the initial guess is V_0, then the first iterate solution is V_1, which is far away from the exact solution V^*. Because the derivative is large when V_k is negative, so it requires a large number of iteration to converge to V^*. Fig. 3.15(b) and (c) illustrate the slow convergence problem with a saturated load MOS inverter circuit and a depletion load MOS inverter circuit as shown in Fig. 3.14(b) and (c) respectively.

The above slow convergence problem can be resolved by using the piecewise nonlinear approach. For example, for the resistor load MOS inverter circuit, first, the MOSFET characteristic is partitioned into two pieces as shown in Fig. 3.16, the first piece is from $-\infty$ to zero, the second piece is from zero to $+\infty$, then the circuit can be solved as illustrated in Fig. 3.16. The initial guess V_0 is in piece II, the first iterate solution $\hat{V}_1$ is in piece I, so V_1 is chosen to be the breakpoint zero. Since $\hat{V}_2$ is within piece II, the Newton-Raphson method is used to find the solution V^*.

Remark: In the iteration scheme for MOS circuits, the piecewise nonlinear approach is used for V_{DS}. The change of V_{GS} and V_{GD} are limited by 1V at each iteration.

3.7. Discussion

Two large circuits were used to test the piecewise nonlinear approach, one is a bipolar circuit as shown in Fig. 3.17, the other is a MOS circuit as shown in Fig.3.18. The data are given in Table 3.2. The results show that this method improves the convergence property of the basic

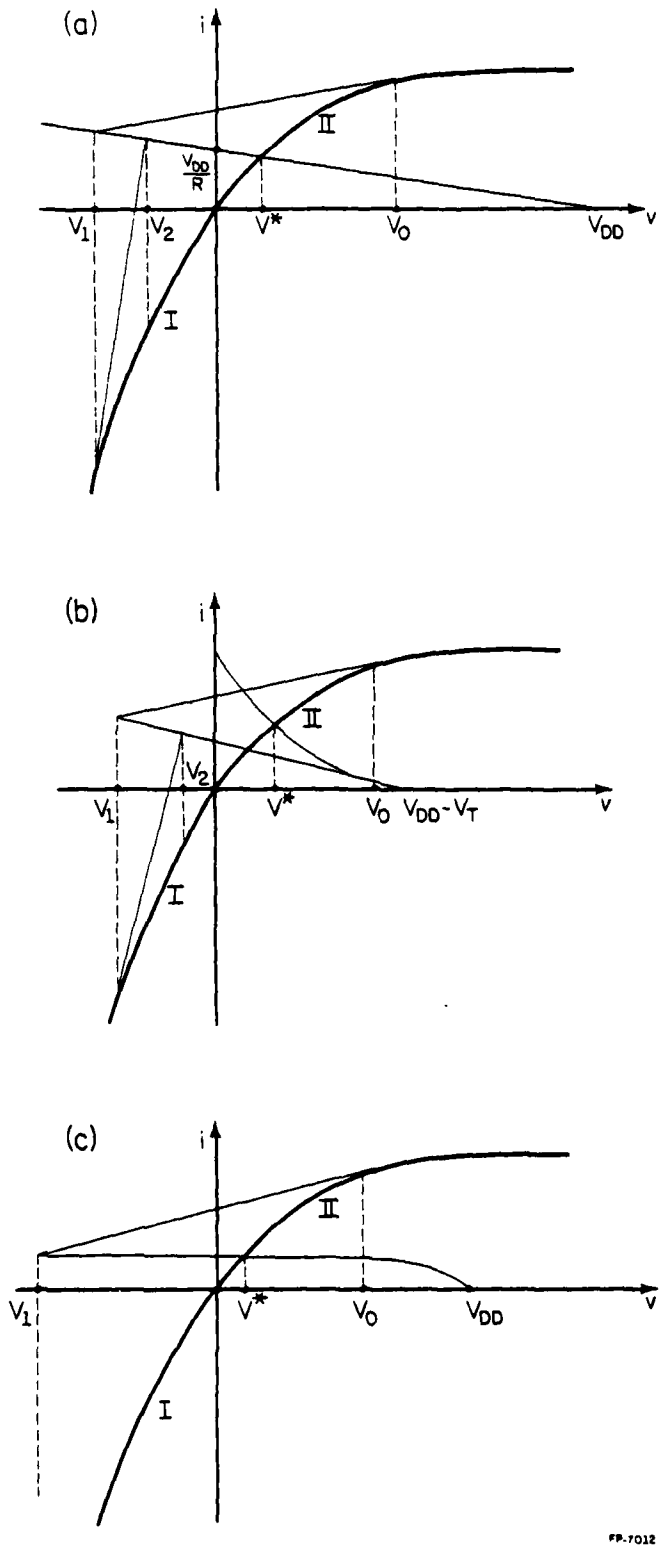
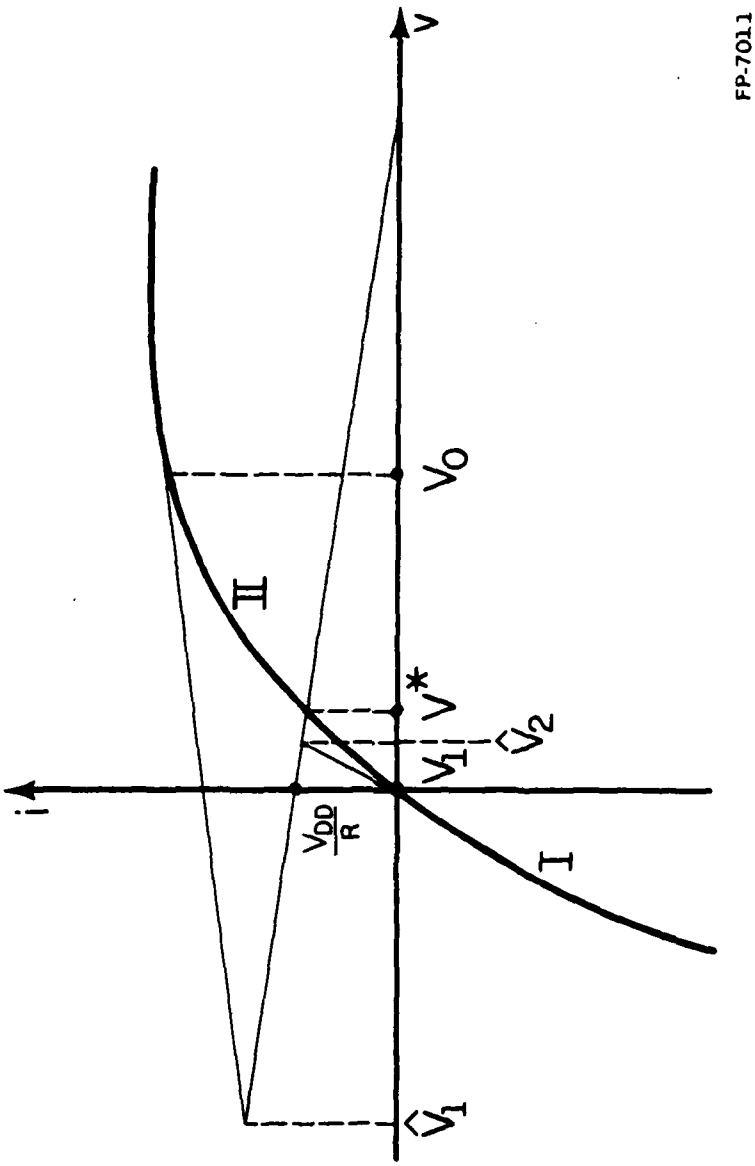
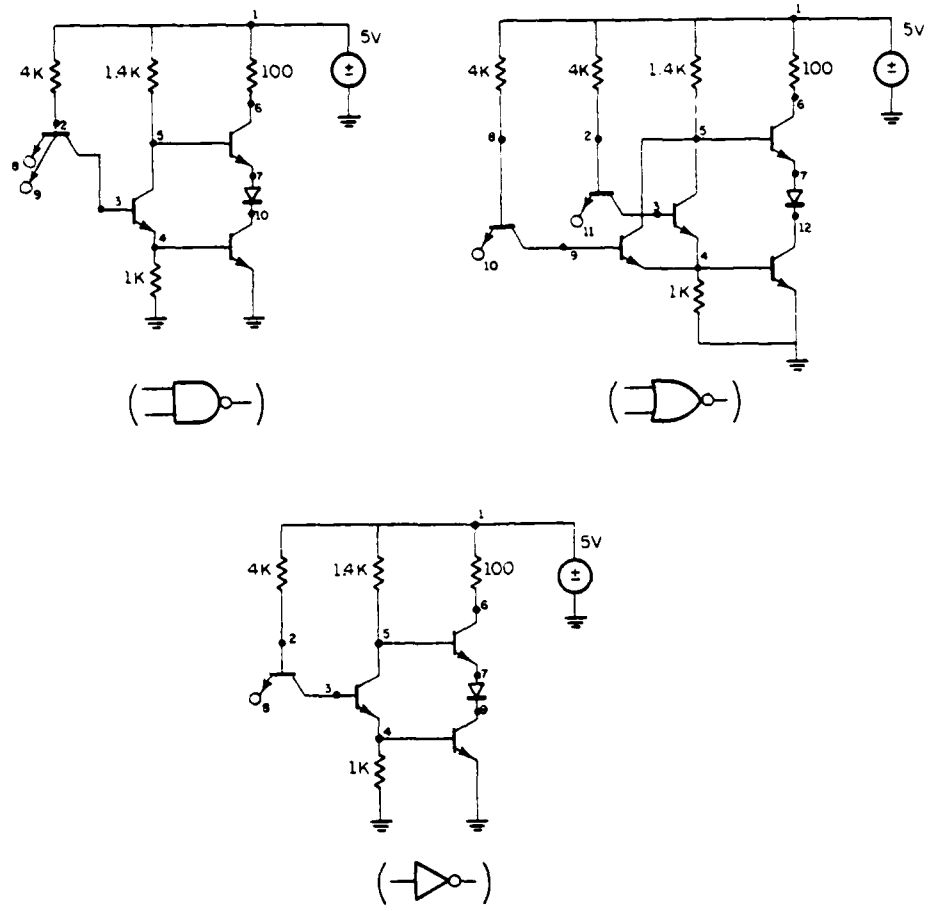


Fig. 3.15 The Slow Convergence Problem with the Circuits in Fig. 3.14.



FP-7011

Fig. 3.16 Example of the Piecewise Nonlinear Approach for the Resistor Load MOS Inverter Circuit.



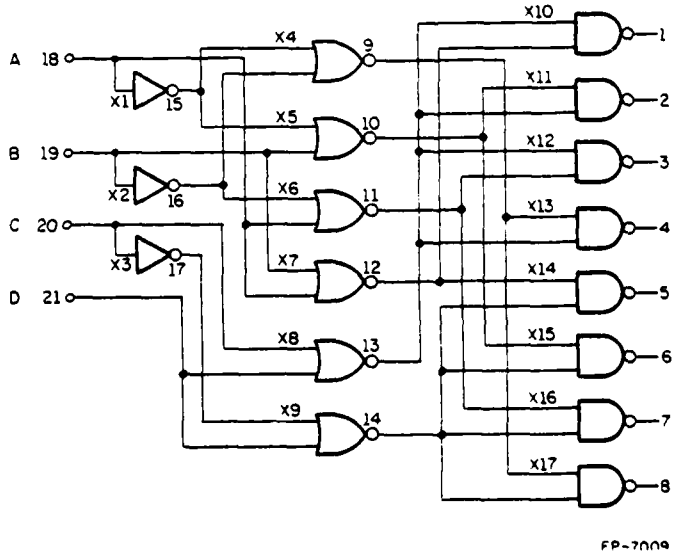


Fig. 3.17 Binary-to-Octal Decoder.

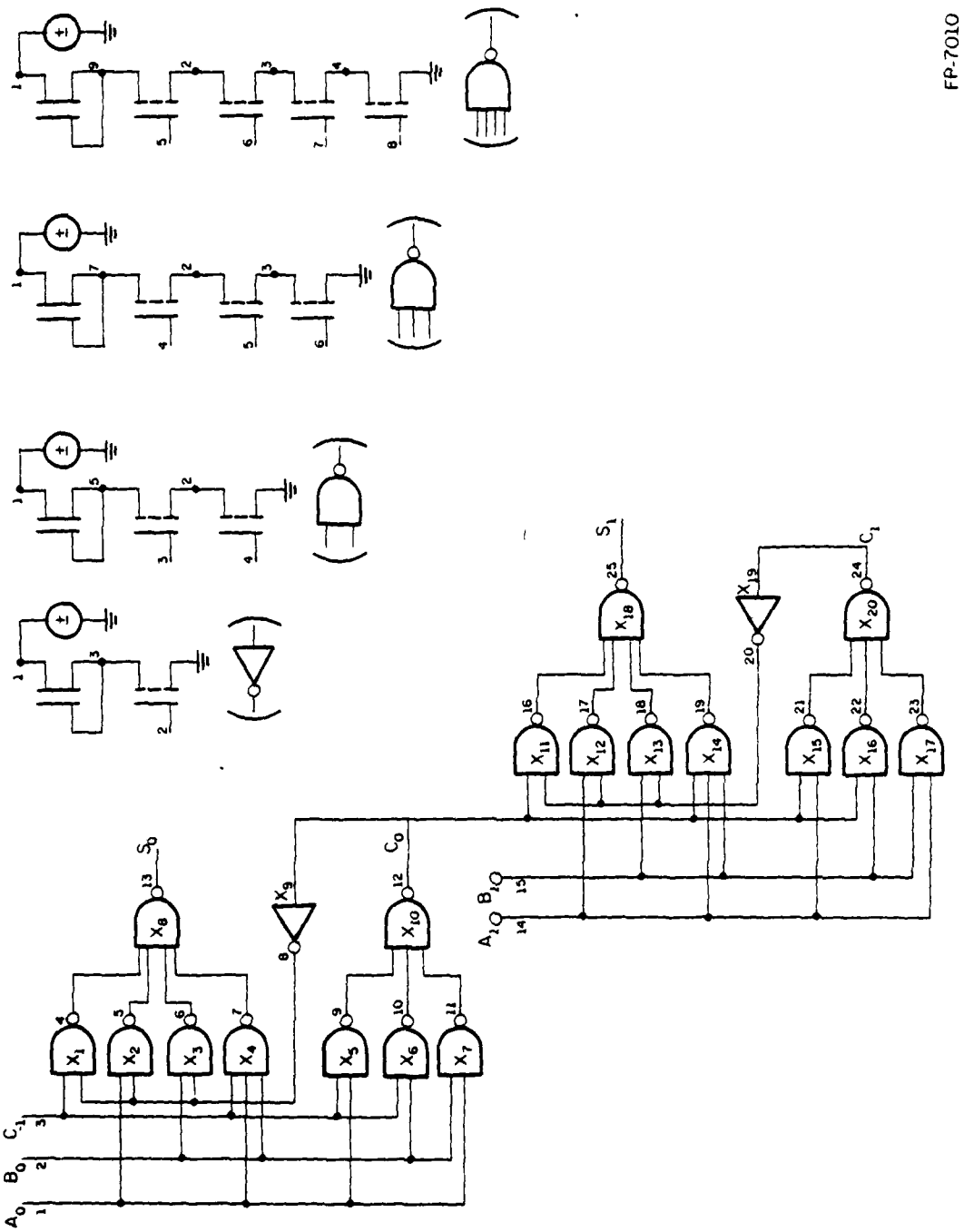


Fig. 3.18 2-Bit Full Adder.

Table 3.2 Comparison of the Results between the Piecewise Nonlinear Approach (PWNL) and SPICE2.

Circuit	Number of iterations (PWNL)	Number of iterations (SPICE2)
Fig. 3.17	34	48
Fig. 3.18	10	28

Newton-Raphson method; however, the proof of the global convergence property or the conditions for global convergence have not yet been derived. More research work on this topic is needed.

IV. NUMERICAL INTEGRATION

A numerical integration method is required to determine the transient response of a circuit. In order to make the numerical integration more accurate and efficient, some method of dynamically varying the timestep is needed, this is usually accomplished by a local truncation error (LTE) timestep control.

Let us denote the upperbound on the local truncation error by ET. In previous work, ET was established as follows [36]. First, a maximum allowable global truncation error GE_{max} and the solution interval T are specified. An assumption that this global error is distributed uniformly within T is made, then the maximum allowable ET per timestep (h) is given by

$$ET = \frac{GE_{max}}{T} * h \tag{4.1}$$

The LTE timestep control with trapezoidal integration is implemented as follows. First, the timestep h and $t_{n+1} = t_n + h_n$ are determined, the solution at the timepoint t_{n+1} is found, then the local truncation error (LTE) is evaluated by Eq. (4.2).

$$LTE = \frac{h^3}{12} * \dddot{x}(\tau) \approx \frac{h^3}{2} DD3 \tag{4.2}$$

where DD3 is the 3rd divided difference [1] and $t_n \leq \tau \leq t_{n+1}$. The kth divided difference is defined by the recursive relation

$$DDk = \frac{DD_{k-1}(t_{n+1}) - DD_{k-1}(t_n)}{\sum_{i=1}^{k} h_{n+1-i}} \quad , \quad DD1 = \frac{x(t_{n+1}) - x(t_n)}{h_n} \tag{4.3}$$

If LTE > ET, then the timestep is considered too large, h_n is rejected, a new h_n is computed using

$$h_n = \sqrt[2]{\frac{2*GE_{max}}{T*DD3}} \tag{4.4}$$

and then a new timepoint t_{n+1} is determined. If, on the other hand, LTE < ET, then the local truncation error at timepoint t_{n+1} is considered satisfactory, and the timestep h_{n+1} is computed using

$$h_{n+1} = \sqrt[2]{\frac{2*GE_{max}}{T*DD3}} \tag{4.5}$$

4.1. Problems with Previous Work

When the above strategy is applied to determine the transient response of a circuit, there are two problems:

(1) Since DD3 is only an approximation of $\dddot{x}(\tau)$, whenever the timestep is changed or the input signal changes abruptly, our investigation shows that DD3 becomes an inaccurate estimate of LTE. This inaccuracy results in the following unwanted situations. One situation is that at the timepoint t_{n+1}, if LTE < ET, the timestep h_{n+1} is increased, but at the next timepoint t_{n+2}, due to the inaccuracy, LTE is now found to be larger than ET, so this timepoint is rejected and the timestep is reduced. The other situation is even worse. If, at the timepoint t_n, the input changes abruptly, then due to the inaccuracy of the DD3 approximation to $\dddot{x}(\tau)$, LTE may be greater than ET and the timestep is reduced. Sometimes this happens repeatedly until the timestep becomes too small and the program terminates. These two situations are explained in detail later.

(2) For digital circuits, the total solution time T may consist of several switching intervals. If a stable numerical integration method is used, initially in an interval the local truncation error accumulates and the global truncation error (GE) increases, but as the solution nears

steady state in a given switch interval the global error decreases, and as the solution approaches the steady state the global error goes to zero, so that the upperbound ET given by Eq. (4.1) is too conservative. This is illustrated in Fig. 4.1.

Now we will consider the above two problems in more detail. The first situation of the first problem can be illustrated by a simple RC circuit as shown in Fig. 4.2. In order to simplify the analysis, the backward Euler method is used. The exact solution for this circuit is

$$v(t) = 5e^{-t/\tau} \tag{4.6}$$

the solution obtained by the backward Euler method is

$$v_{n+1} = \frac{v_n}{1 + h_{n/\tau}} \tag{4.7}$$

where $v_o = 5V$.

The local truncation error estimates at timepoints t_{n+1} and t_n are

$$LTE_{n+1} = h_n^2 * DD2_{n+1} \tag{4.8}$$

$$LTE_n = h_{n-1}^2 * DD2_n \tag{4.9}$$

where $$DD2_{n+1} = \frac{\frac{v_{n+1} - v_n}{h_n} - \frac{v_n - v_{n-1}}{h_{n-1}}}{h_n + h_{n-1}}$$

$$DD2_n = \frac{\frac{v_n - v_{n-1}}{h_{n-1}} - \frac{v_{n-1} - v_{n-2}}{h_{n-2}}}{h_{n-1} + h_{n-2}}$$

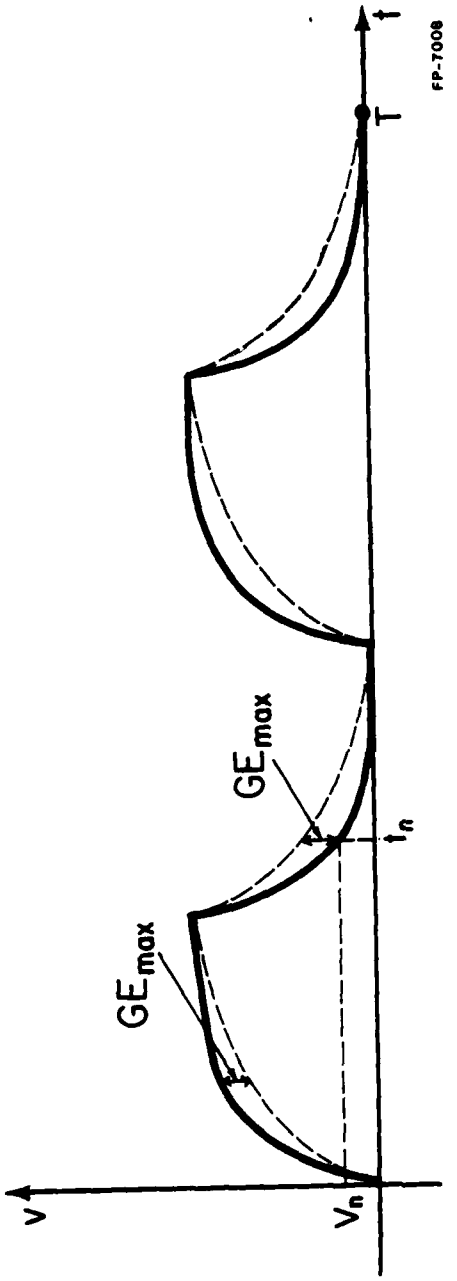
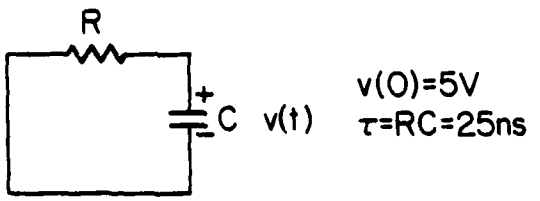
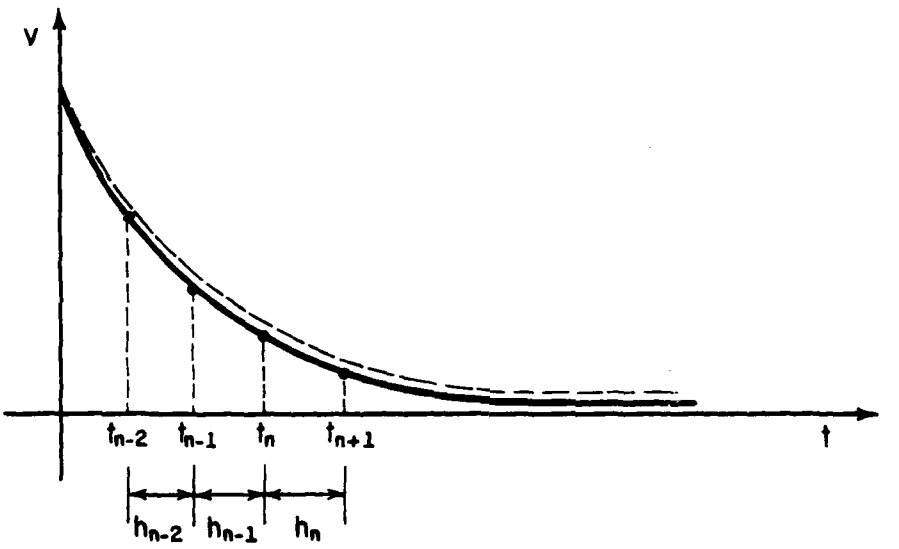


Fig. 4.1 Numerical Integration Solution v.s. Exact Solution.



v(0)=5V
τ=RC=25ns

(a)



(b)

FP-7007

Fig. 4.2(a) Simple RC Circuit.
(b) Waveform of the Simple RC Circuit.

Let us consider the situation when $h_{n-2} = h_{n-1} = h$ and $h_n = ah$, where a is a ratio constant. From Eqs. (4.7), (4.8) and (4.9), we obtain

$$\frac{LTE_{n+1}}{LTE_n} = \frac{DD2_{n+1}}{DD2_n} \frac{h_n^2}{h_{n-1}^2} = \frac{2a}{a+1} * a^2 * [\frac{1}{1 + \frac{ah}{\tau}}] \tag{4.10}$$

If both LTE_{n+1} and LTE_n are good approximations of the true local truncation errors, then from Eq. (4.6), (4.7) and the definition of local truncation error [36] the ratio of LTE_{n+1} over LTE_n should be

$$\frac{LTE_{n+1}}{LTE_n} \approx a^2 * [\frac{1}{1+\frac{ah}{\tau}}] \tag{4.11}$$

Comparison of Eq. (4.11) with Eq. (4.10) shows that the ratio computed by Eq. (4.10) is wrong by a factor of $\frac{2a}{a+1}$. When a = 1, that is, the timestep is constant, then the estimation by Eq. (4.8) is good. When a is different from 1, then the estimation by Eq. (4.8) is not good. Table 4.1 gives the simulation results of the simple RC circuit, which confirms the above conclusion. The ET used is 10^{-3}V. At the first three timepoints, the timesteps are kept constant, so the estimation by Eq. (4.8) is good. At the fourth timepoint the timestep is increased by a factor of two. The true local truncation error is 0.8930E-3, which is an acceptable error; but the estimation by Eq. (4.8) is 0.1207E-2, which is larger than ET, so the timepoint is rejected.

Now we would like to see if this inaccuracy can be explained by the above conclusion. The ratio of the estimation at the fourth timepoint over the estimation at the third timepoint is

Table 4.1 Simulation Results of the Simple RC Circuit.

t (ns)	h(timestep)	LTE(estimate)	True local truncation error
1.5	0.25	0.2355E-3	0.2339E-3
1.75	0.25	0.2332E-3	0.2314E-3
2.00	0.25	0.2309E-3	0.2293E-3
2.50	0.50	0.1207E-2	0.8930E-3

$$\frac{0.1207E-2}{0.2309E-3} = 5.227 \tag{4.12}$$

instead of 4 as predicted by Eq. (4.11). However, note that for a = 2

$$\frac{2a}{a+1} = \frac{4}{3} = 1.333 \approx \frac{5.227}{4} \tag{4.13}$$

Eq. (4.13) shows that the estimation of local truncation error is really wrong by a factor of $\frac{2a}{a+1}$ as shown in Eq. (4.10).

Now let us consider the second situation of the first problem. This situation can be illustrated by a simple RC circuit as shown in Fig. 4.3. Again the backward Euler method is used here for the simplicity of the analysis. The exact solution for this circuit is

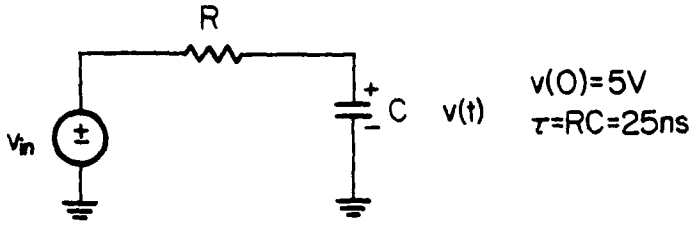
$$v(t) = \begin{cases} 5e^{-t/\tau} & t \leq t_n \\ v_n e^{-t/\tau} + 5(1 - e^{-(t-t_n)/\tau}) & t > t_n \end{cases} \tag{4.14}$$

the solution obtained by backward Euler method is

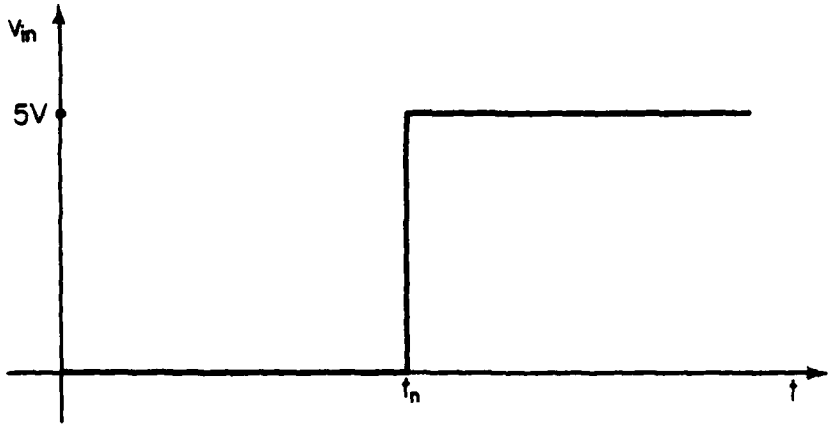
$$v_{k+1} = \begin{cases} \dfrac{v_k}{1+h_{k/\tau}} & k < n \\ \dfrac{v_k}{1+h_{k/\tau}} + \dfrac{5\frac{h_k}{\tau}}{1+h_{k/\tau}} & k \geq n \end{cases} \tag{4.15}$$

where $v_o = 5V$.

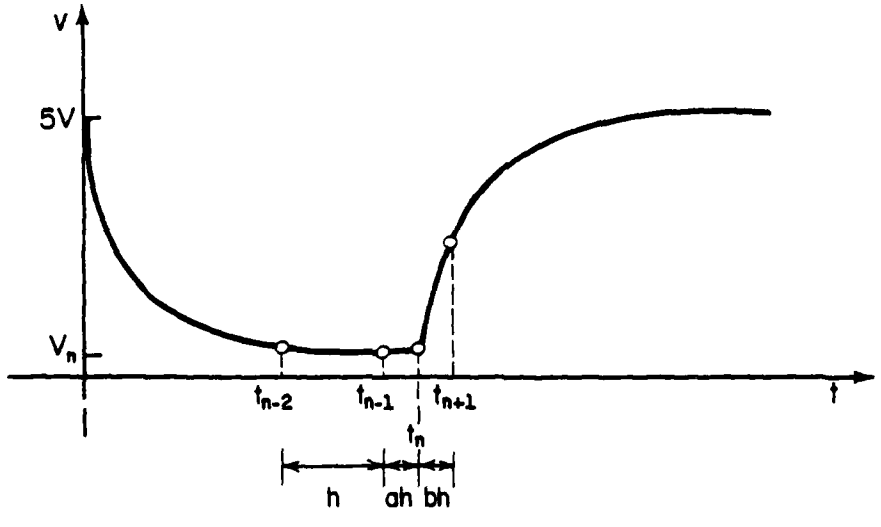
In the early version of SPICE2, when t_n exceeded a source breakpoint, then h_{n-1} was reduced such that the value t_n coincides with the breakpoint. The timestep was reduced to a small value and then the iteration was



(a)



(b)



(c)

FP-7027

Fig. 4.3(a) Simple RC Circuit.
(b) Waveform of v_{in}.
(c) Waveform of v.

continued. Let us consider the situation when $h_{n-2} = h$, $h_{n-1} = ah$ and $h_n = bh$, where a and b are ratio constants. From Eqs. (4.8) and (4.15), we obtain the following estimate of the local truncation error:

$$LTE_{n+1} = b^2h^2 \left[\frac{5}{\tau(a+b)h} + \frac{b(v_{n+1} - 5)}{\tau^2(a+b)}\right] \tag{4.16}$$

Let us also assume that bh is not small enough, so that

$$LTE_{n+1} > ET \tag{4.17}$$

Then the timestep was reduced and a new timestep ch was computed by

$$c^2h^2\left[\frac{5}{\tau(a+b)h} + \frac{b(v_{n+1} - 5)}{\tau^2(a+b)}\right] = ET \tag{4.18}$$

where $c < b$.

The local truncation error for this new timestep ch is estimated by

$$LTE'_{n+1} = c^2h^2\left[\frac{5}{\tau(a+c)h} + \frac{c(v_{n+1} - 5)}{\tau^2(a+c)}\right] \tag{4.19}$$

It follows that

$$\frac{LTE'_{n+1}}{ET} = \frac{a+b}{a+c} \frac{\left[1 + \frac{ch(v_{n+1} - 5)}{5\tau}\right]}{\left[1 + \frac{bh(v_{n+1} - 5)}{5\tau}\right]} \tag{4.20}$$

Since $c < b$ then $(a + c) < (a + b)$ and for a small enough a the ratio in Eq. (4.20) could be greater than one. If this is the case, then the step

size will be reduced again. This could happen again and again. This was the case in early version of the SPICE2 program, and frequently after abrupt clock or signal changes the program would not converge and the job would be terminated prematurely.

Remark: In Eq. (4.16), the second term is an approximation to the true local truncation error, the first term, although it is dominant, is a parasitic term which is generated by the use of voltages at the timepoints of the previous switching interval.

The second problem is detailed as follows. Let GE_{max} denote the maximum global truncation error at t_n (Fig. 4.1). Assume that the trapezoidal method is used and that we are dealing with an exponentially decaying waveform. The local truncation error at the timepoint t_{n+1} is given by

$$LTE_{n+1} = \frac{h_n^3}{12} \dddot{x}(t') \qquad t_n \le t' \le t_{n+1} \tag{4.21}$$

and for this example

$$LTE_{n+1} \approx \frac{h_n^3}{12\tau^3} V_{n,max} \tag{4.22}$$

From Eq. (4.22) and the definition of local truncation error, we obtain

$$GE_{n+1} \approx GE_{max}\, e^{-h_n/\tau} + \frac{h_n^3}{12\tau^3} V_{n,max} \le GE_{max} \tag{4.23}$$

where GE_{n+1} is the global truncation error at the timepoint t_{n+1}. Eq. (4.23) can be reduced to

$$\frac{h_n^3}{12\tau^3} V_{n,max} \le GE_{max}\left(\frac{h_n}{\tau}\right) \tag{4.24}$$

The local truncation error timestep control requires that

$$LTE_{n+1} = \frac{h_n^3}{12\tau^3} V_{n,max} \leq ET \tag{4.25}$$

Assuming equality in Eq. (4.25) and eliminating h_n/τ in Eq. (4.24), we obtain the following upper bound on the local truncation error.

$$ET \leq \sqrt{\frac{12(GE_{max})^3}{V_{n,max}}} \tag{4.26}$$

In order to check the above bound which was derived for RC circuits, a number of digital circuits were simulated and the following empirical bound on the local truncation error was determined.

$$ET = 10 \sqrt{\frac{(GE_{max})^3}{V_{DD}}} \tag{4.27}$$

where V_{DD} is the voltage swing which is the supply voltage in this example. Eq. (4.27) also holds for exponentially rising waveforms. Given GE_{max} and V_{DD}, then ET can be determined by Eq. (4.27), and then Eq. (4.2) can be used to control the timestep.

Eq. (4.26) shows that ET is proportional to $(GE_{max})^{3/2}$ for RC circuits if the trapezoidal method is used. In general, for RC circuits, if a stable numerical integration method of order n is used, similar derivation as used above can show that

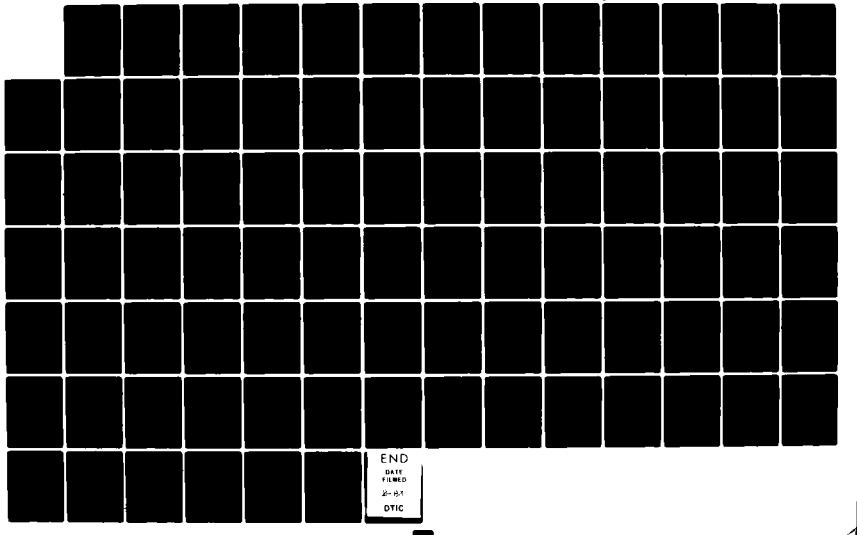
$$ET \propto (GE_{max})^{(n+1)/n} \tag{4.28}$$

Eq. (4.28) was verified experimentally for the backward Euler method and the trapezoidal method. The RC circuit as shown in Fig. 4.2 was used. The simulation results are given in Figs. 4.4 and 4.5.

AD-A124 064 AN INVESTIGATION OF ORDERING TEARING AND LATENCY ALGORITHMS FOR THE TIME-...(U) ILLINOIS UNIV AT URBANA COORDINATED SCIENCE LAB P YANG AUG 80 R-891

2/2

UNCLASSIFIED N000014-79-C-0424 F/G 9/5 NL

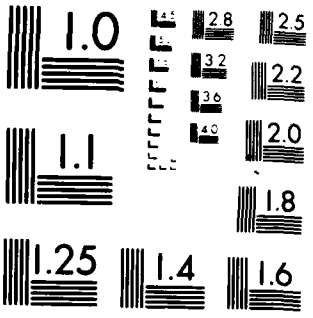


END

DATE FILMED

4-84

DTIC



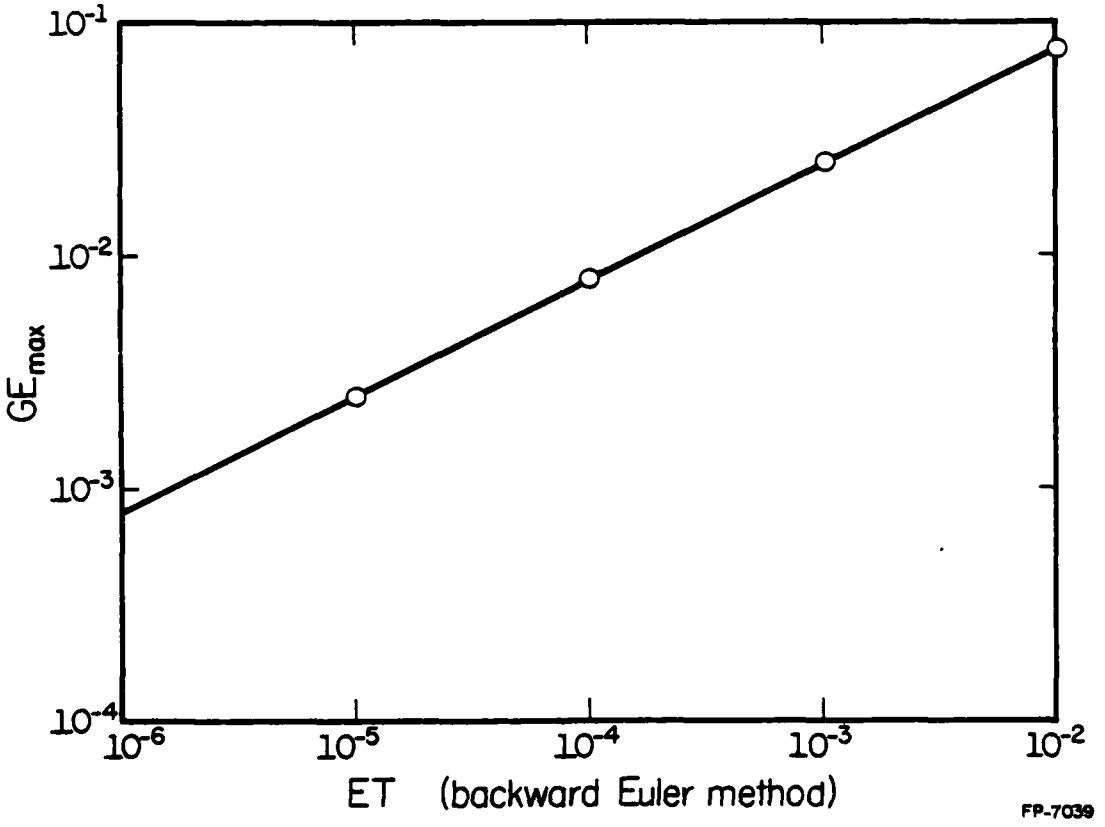


Fig. 4.4 Simulation Data of the Backward Euler Method Which show that ET $\propto (GE_{max})^2$.

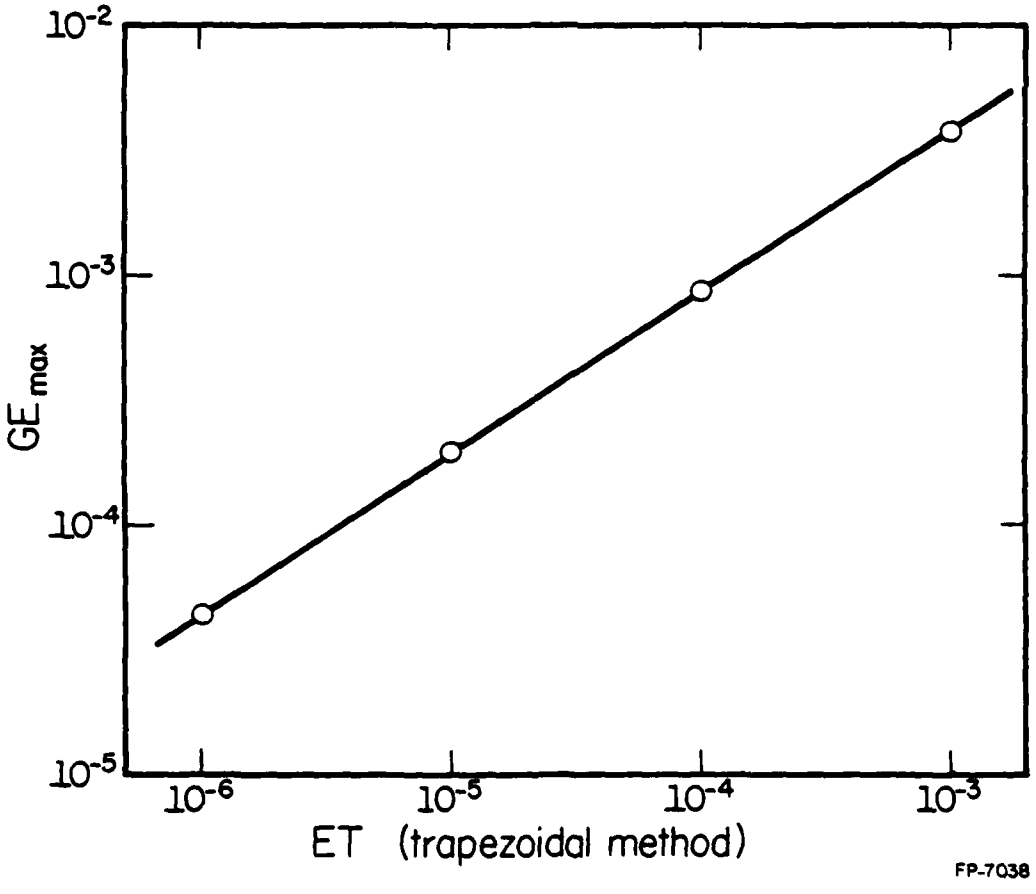


Fig. 4.5 Simulation Data of the Trapezoidal Method Which Show that $ET \propto (GE_{max})^{3/2}$.

4.2. Algorithm

The following algorithm for variable timestep control was developed based on the discussion in the previous section. The trapezoidal method is assumed. ET is computed by Eq. (4.27).

First let us derive an expression for the estimation of the local truncation error which is used in the algorithm. Let us assume that $h_{n-2} = h$, $h_{n-1} = ah$ and $h_n = bh$. The solution obtained by the trapezoidal method for an exponentially decaying waveform is

$$v_{n+1} = v_n \frac{1 - h_n/2\tau}{1 + h_n/2\tau} \tag{4.29}$$

the 3rd divided difference is given by

$$DD3_{n+1} = \frac{DD2_{n+1} - DD2_n}{h_{n-2} + h_{n-1} + h_n}$$

$$= \frac{3(1+b)}{2(1+a+b)} * \frac{-v_n}{6\tau^3(1 - \frac{h}{2\tau})(1 - \frac{ah}{2\tau})(1 + \frac{bh}{2\tau})} \tag{4.30}$$

where $\frac{3(1+b)}{2(1+a+b)}$ is the error factor in the estimate of the third derivative caused by varying the timestep.

By taking into account the effect of different timesteps, the expression of local truncation error is given by

$$LTE_{n+1} = \frac{h_n^3}{2} * DD3_{n+1} * \frac{2(1+a+b)}{3(1+b)} \tag{4.31}$$

or we can define a new quantity DD3' which is given by

$$DD3'_{n+1} = \frac{DD2_{n+1} - DD2_n}{(h_{n-2} + h_n)} \tag{4.32}$$

then Eq. (4.31) can be reduced to

$$LTE_{n+1} = \frac{h_n^3}{3} * DD3'_{n+1} \tag{4.33}$$

Algorithm:

(1) Record the initial time t_0, final time t_f, minimum stepsize h_{min}, maximum stepsize h_{max}, and source breakpoints.

(2) Set the initial timestep $h = h_{min}$.

(3) Compute X_1 at $t_1 = t_0 + h$, X_2 at $t_2 = t_0 + 2h$, and X_3 at $t_3 = t_0 + 3h$.

(4) Set $n = 3$ and compute LTE by Eq. (4.33).

(5) Compute $h_n = h\sqrt[3]{\frac{ET}{LTE}}$. If $h_n < 0.6h$, then $h = h_n$ and go to (3); otherwise, continue.

(6) Compute $t_{n+1} = t_n + h_n$. If t_{n+1} does not exceed a source breakpoint, then go to (7). If t_{n+1} exceeds a source breakpoint, then h_n is reduced such that the value t_{n+1} coincides with the breakpoint. Compute X_{n+1} for this breakpoint. Compute LTE by Eq. (4.33), compute h_{n+1}, if $h_{n+1} < 0.6h_n$, then $h_n = h_{n+1}$ and go to (6); otherwise, set $h = h_{min}$ and $t_0 = t_{n+1}$, then go to (3).

(7) Compute X_{n+1}. Compute LTE by Eq. (4.33), compute h_{n+1}, if $h_{n+1} < 0.6h_n$, then $h_n = h_{n+1}$ and go to (6); otherwise, continue.

(8) If $t_{n+1} > t_f$, then stop; if not, then n = n+1, and go to (6).

Remark: The above algorithm has been derived for a fixed order variable stepsize method which uses the trapezoidal rule. Our simulation results show that the problems we mentioned before in this chapter are resolved by this algorithm. If other fixed order methods are to be used, then the corresponding equations should be modified.

V. TEARING METHODS AND SPARSITY CONSIDERATIONS FOR NODE TEARING METHOD

There are two kinds of tearing methods - the branch tearing method and the node tearing method. The idea of branch tearing was first introduced by Kron [12]. Recently Chua and Chen [16] have shown that the branch tearing is just a special case of generalized hybrid analysis. The main idea of branch tearing is to select a set of tearing branches first, then the given network is torn apart into several subnetworks by removing these tearing branches (Fig. 5.1), analyzing each subnetwork separately, and obtaining the solution of the entire network by combining the solutions of the subnetworks via the tearing branches. Algebraically, this method is equivalent to a particular ordering of the hybrid analysis equations such that the resulting matrix has a bordered block-diagonal structure (Fig. 5.2). Each block corresponds to a subnetwork, and the border corresponds to the interconnections of the subnetworks.

The idea of node tearing was first introduced by Sangiovanni-Vincentelli, Chen and Chua [20]. The main idea is to select a set of tearing nodes first, then the network is torn apart into several subnetworks by removing these tearing nodes (Fig. 5.3), each subnetwork is analyzed separately, and the solution of the entire network is obtained via the tearing nodes. Algebraically, this method is equivalent to a particular ordering of nodal analysis equations such that the resulting matrix has a bordered block-diagonal structure (Fig. 5.4). Each block corresponds to a subnetwork, and the border corresponds to the interconnections of the subnetworks.

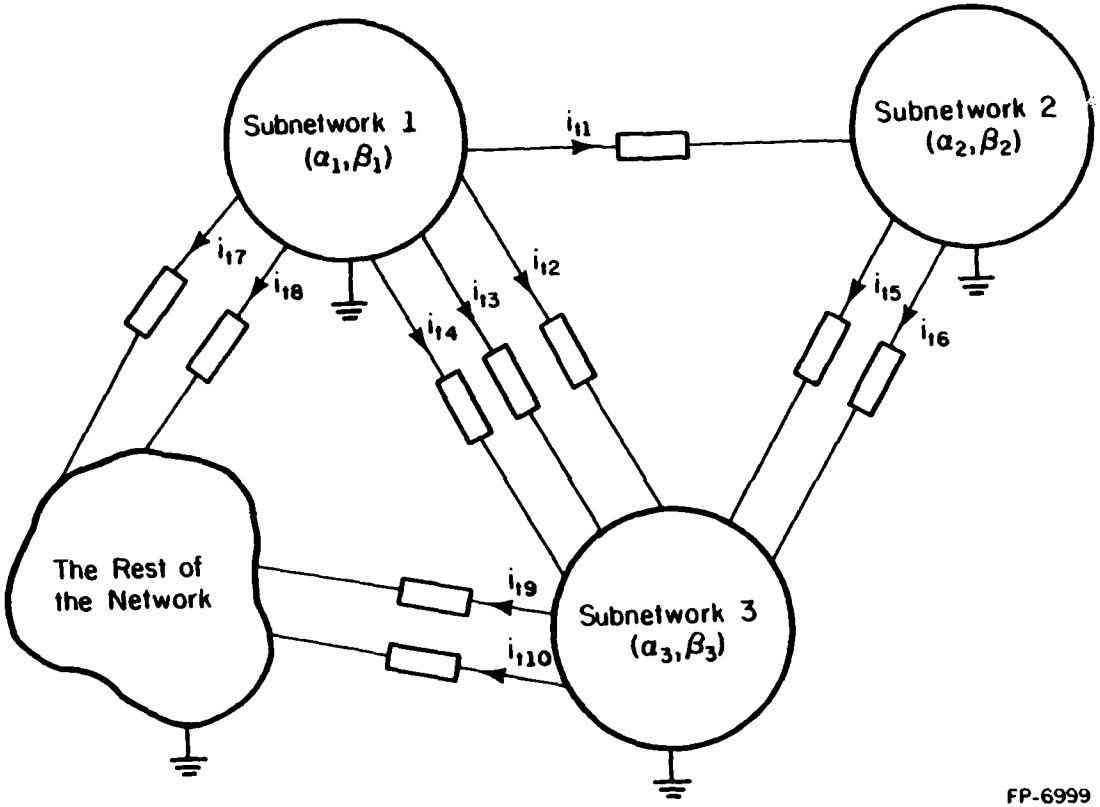


Fig. 5.1 Example of a Network Partitioned into Three Subnetworks by the Branch Tearing Method.

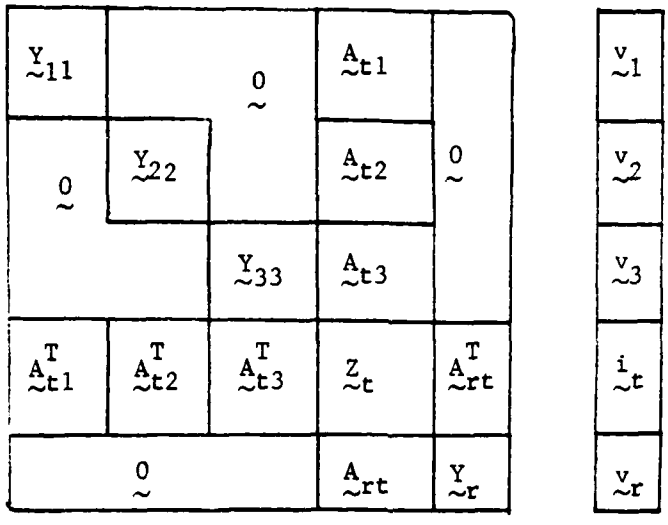


Fig. 5.2 Bordered Block-Diagonal Matrix Formulated by Branch Tearing.

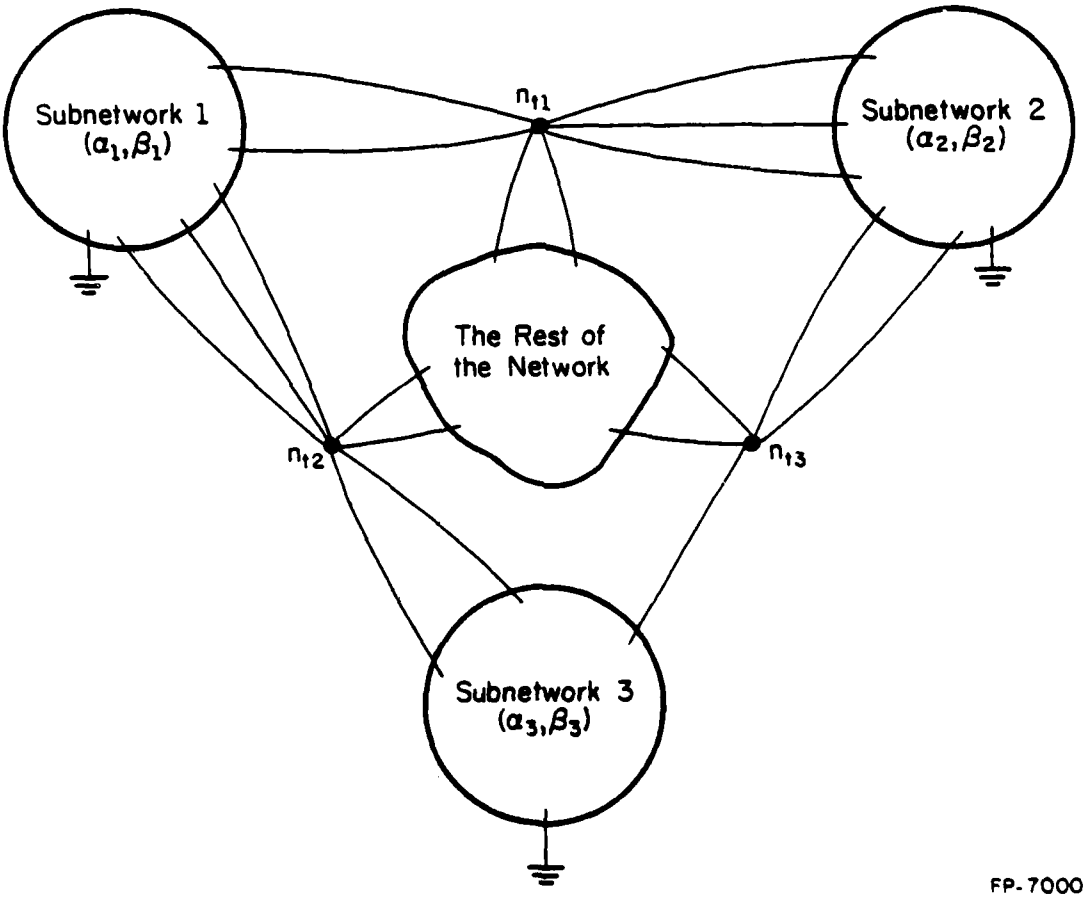


Fig. 5.3 Example of a Network Partitioned into Three Subnetworks by the Node Tearing Method.

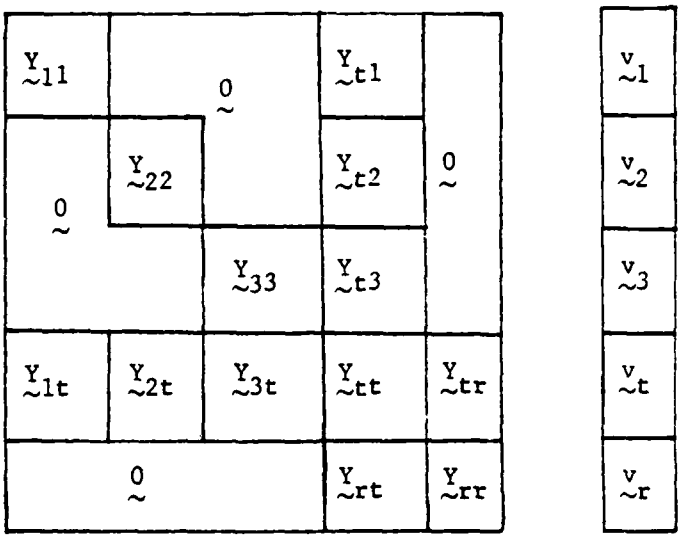


Fig. 5.4 Bordered Block-Diagonal Matrix Formulated by Node Tearing.

Recently, because tearing methods possess several advantages over conventional circuit analysis methods, a lot of effort has been devoted to the study of tearing methods for the analysis of large scale circuits. The advantages of tearing methods are as follows: First, tearing methods are suitable for the exploitation of the repetitiveness of a limited number of subnetworks; secondly, tearing methods are suitable for the exploitation of latency; thirdly, tearing methods are suitable for parallel processing. In order to solve the network by tearing methods one must specify a partitioning strategy, and also one must specify a technique for solving the partitioned equations. In the literature mostly the branch tearing method has been used to solve large networks [22-24]. The solution strategy has been to estimate the current or voltage at each tearing port and to excite the torn subnetworks with independent sources at these ports. The remaining port responses are computed, and are substituted into the interconnection equations. If these equations are not satisfied, then another estimate is made of the variables chosen as port excitations. This iterative procedure continues until convergence is achieved. If the subnetworks are nonlinear a multilevel iteration scheme is used, such as a Gauss-Seidel [22], Newton-SOR [24] or a multilevel Newton iteration [42]. However, the first two iteration schemes do not have second-order convergence while the third scheme requires the computation of an additional Jacobian matrix.

Because the above approach introduces new variables, such as tearing branch currents, the complexity of the problem is increased. Also, a multilevel iteration scheme is required. Another disadvantage of the branch tearing method is that each subnetwork must contain the datum node for the network, or else a local datum node must be chosen for each subnetwork. In the program SLATE a different approach is used, and the

internal subnetwork variables are eliminated from the tearing node equations. Then the tearing node voltages are computed. Only a one level Newton iteration is required, and the internal variables for each subnetwork can be eliminated using parallel processing methods. The elimination of the internal variables is equivalent to replacing the subnetworks by Norton equivalent circuits at the tearing nodes.

In our study of tearing methods it was assumed that the user specifies the subnetworks, and any part of the network not specified as a subnetwork is automatically included in the subnetwork called rest of network in Figs. 5.1 and 5.3. The subnetworks are processed first in the solution algorithm, and the tearing branches or nodes along with the rest of network equations are processed last. Thus, the branch tearing method and the node tearing method described in Section 5.1 and Section 5.2 are somewhat different from those found in the literature [12-14]. In Section 5.3, a comparison between branch tearing and node tearing is given. In Section 5.4, the derivation of the construction of the node tearing matrix from subnetworks is detailed. The sparsity considerations for the node tearing method are presented in Section 5.5. The implementation of node tearing is described in Section 5.6. The circuit interpretation of the tearing methods is given in Section 5.7. Some conclusions are given in Section 5.8.

5.1. Derivation of the Branch Tearing Method

Let N be a connected network having (n+1) nodes: the datum node n_0 and the nondatum nodes $\alpha = \{n_1, n_2, \ldots\ldots, n_n\}$, and b branches, $\beta = \{b_1, b_2, \ldots\ldots b_b\}$. Let the branch voltages, branch currents and the node-to-datum voltages be denoted by $\underset{\sim}{e} = (e_1, e_2, \ldots\ldots e_b)^T$,

$\underset{\sim}{i} = (i_1, i_2, \ldots\ldots i_b)^T$ and $\underset{\sim}{v} = (v_1, v_2, \ldots\ldots v_n)^T$, respectively. Let the interconnection be defined as the remaining part of the network when all the subnetworks are removed, that is, the set of tearing branches and the rest of the network in Fig. 5.1. Let us assume that a proper set of tearing branches has been chosen such that there is no mutual coupling either among the torn subnetworks or between the torn subnetworks and the interconnection, and that all subnetworks contain the common datum node. This latter assumption is made in order to avoid floating subnetworks which result in singular submatrices. The more general case when all subnetworks do not contain a common datum node is discussed in [17]. Subscripts s, t and r are used to denote quantities pertaining to the subnetworks, the tearing branches and the remaining branches, respectively, so the branch set β is partitioned into three subsets β_s, β_t and β_r (Fig. 5.1), and the node set α is partitioned into two subsets α_s and α_r. This yields the following special structures for the reduced incidence matrix $\underset{\sim}{A}$ and the branch conductance matrix $\underset{\sim}{G}$ of network N:

$$\underset{\sim}{A} = \begin{array}{c} \\ \alpha_s \\ \alpha_r \end{array} \begin{array}{c} \begin{array}{ccc} \beta_s & \beta_t & \beta_r \end{array} \\ \begin{bmatrix} \underset{\sim}{A}_s & \underset{\sim}{A}_t & \underset{\sim}{0} \\ \underset{\sim}{0} & \underset{\sim}{A}_{rt} & \underset{\sim}{A}_{rr} \end{bmatrix} \end{array} \tag{5.1}$$

$$\underset{\sim}{G} = \begin{array}{c} \\ \beta_s \\ \beta_t \\ \beta_r \end{array} \begin{array}{c} \begin{array}{ccc} \beta_s & \beta_t & \beta_r \end{array} \\ \begin{bmatrix} \underset{\sim}{G}_s & \underset{\sim}{0} & \underset{\sim}{0} \\ \underset{\sim}{0} & \underset{\sim}{G}_t & \underset{\sim}{0} \\ \underset{\sim}{0} & \underset{\sim}{0} & \underset{\sim}{G}_r \end{bmatrix} \end{array} \tag{5.2}$$

Let the torn network have k subnetworks $N_1, N_2, \ldots\ldots, N_k$. Let α_s and β_s be partitioned correspondingly into k subsets $\alpha_1, \alpha_2, \ldots\ldots, \alpha_k$ and $\beta_1, \beta_2, \ldots\ldots, \beta_k$ respectively. With this partitioning, the node-to-branch incidence matrix $\underset{\sim}{A}$ can be written as

$$\underset{\sim}{A} = \begin{array}{c|cccc|cc} & \beta_1 & \beta_2 & \cdots & \beta_k & \beta_t & \beta_r \\ \alpha_1 & \underset{\sim}{A}_{s1} & & & & \underset{\sim}{A}_{t1} & \\ \alpha_2 & & \underset{\sim}{A}_{s2} & & \underset{\sim}{0} & \underset{\sim}{A}_{t2} & \\ \vdots & & \ddots & & & \vdots & \underset{\sim}{0} \\ & & \underset{\sim}{0} & \ddots & & & \\ \alpha_k & & & & \underset{\sim}{A}_{sk} & \underset{\sim}{A}_{tk} & \\ \hline \alpha_r & & \underset{\sim}{0} & & & \underset{\sim}{A}_{rt} & \underset{\sim}{A}_{rr} \end{array} \qquad (5.3)$$

the branch conductance matrix $\underset{\sim}{G}$ can be written as:

$$\underset{\sim}{G} = \begin{array}{c|cccc|cc} & \beta_1 & \beta_2 & \cdots & \beta_k & \beta_t & \beta_r \\ \beta_1 & \underset{\sim}{G}_{s1} & & & & & \\ \beta_2 & & \underset{\sim}{G}_{s2} & & \underset{\sim}{0} & & \\ \vdots & & \ddots & & & \underset{\sim}{0} & \underset{\sim}{0} \\ & & \underset{\sim}{0} & \ddots & & & \\ \beta_k & & & & \underset{\sim}{G}_{sk} & & \\ \hline \beta_t & & \underset{\sim}{0} & & & \underset{\sim}{G}_{t} & \underset{\sim}{0} \\ \hline \beta_r & & \underset{\sim}{0} & & & \underset{\sim}{0} & \underset{\sim}{G}_{r} \end{array} \qquad (5.4)$$

The network variables are constrained by the Kirchhoff's current law (KCL), Kirchhoff's voltage law (KVL) and the branch constraint relations (BC) [28].

$$\text{(KCL)} \quad \underset{\sim}{A}_{s}\underset{\sim}{i}_{s} + \underset{\sim}{A}_{t}\underset{\sim}{i}_{t} = \underset{\sim}{0} \tag{5.5}$$

$$\underset{\sim}{A}_{rt}\underset{\sim}{i}_{t} + \underset{\sim}{A}_{rr}\underset{\sim}{i}_{r} = \underset{\sim}{0} \tag{5.6}$$

$$\text{(KVL)} \quad \underset{\sim}{e}_{s} = \underset{\sim}{A}^{T}_{s}\underset{\sim}{v}_{s} \tag{5.7}$$

$$\underset{\sim}{e}_{t} = \underset{\sim}{A}^{T}_{ts}\underset{\sim}{v}_{s} + \underset{\sim}{A}^{T}_{rt}\underset{\sim}{v}_{r} \tag{5.8}$$

$$\underset{\sim}{e}_{r} = \underset{\sim}{A}^{T}_{rr}\underset{\sim}{v}_{r} \tag{5.9}$$

$$\text{(BC)} \quad \underset{\sim}{i}_{s} = \underset{\sim}{I}_{ss} + \underset{\sim}{G}_{s}\underset{\sim}{e}_{s} - \underset{\sim}{G}_{s}\underset{\sim}{e}_{ss} \tag{5.10}$$

$$\underset{\sim}{e}_{t} = \underset{\sim}{e}_{ts} + \underset{\sim}{Z}_{t}\underset{\sim}{i}_{t} - \underset{\sim}{Z}_{t}\underset{\sim}{i}_{ts} \tag{5.11}$$

$$\underset{\sim}{i}_{r} = \underset{\sim}{I}_{rs} + \underset{\sim}{G}_{r}\underset{\sim}{e}_{r} - \underset{\sim}{G}_{r}\underset{\sim}{e}_{rs} \tag{5.12}$$

Substituting Eqs. (5.7), (5.8) and (5.9) into Eqs. (5.10) and (5.12), and then substituting the results into Eqs. (5.5) and (5.6), we obtain

$$\underset{\sim}{A}_{s}\underset{\sim}{G}_{s}\underset{\sim}{A}^{T}_{s}\underset{\sim}{v}_{s} + \underset{\sim}{A}_{t}\underset{\sim}{i}_{t} = \underset{\sim}{J}_{ss} \tag{5.13}$$

$$\underset{\sim}{A}_{rt}\underset{\sim}{i}_{t} + \underset{\sim}{A}_{rr}\underset{\sim}{G}_{r}\underset{\sim}{A}^{T}_{rr}\underset{\sim}{v}_{r} = \underset{\sim}{J}_{rs} \tag{5.14}$$

Substituting Eq. (5.8) into Eq. (5.11), we obtain

$$\underset{\sim}{A}^{T}_{ts}\underset{\sim}{v}_{s} - \underset{\sim}{Z}_{t}\underset{\sim}{i}_{t} + \underset{\sim}{A}^{T}_{rt}\underset{\sim}{v}_{r} = \underset{\sim}{E}_{ts} \tag{5.15}$$

where $\underset{\sim}{J}_{ss} \triangleq \underset{\sim}{A}_{s}\underset{\sim}{G}_{s}\underset{\sim}{e}_{ss} - \underset{\sim}{A}_{s}\underset{\sim}{I}_{ss}$,

$$\underset{\sim}{J}_{rs} \triangleq \underset{\sim}{A}_{rr}\underset{\sim}{G}_{r}\underset{\sim}{e}_{rs} - \underset{\sim}{A}_{rr}\underset{\sim}{I}_{rs}$$

and $\underset{\sim}{E}_{ts} \triangleq \underset{\sim}{e}_{ts} - \underset{\sim}{Z}_t \underset{\sim}{i}_{ts}$.

Eqs. (5.13), (5.14) and (5.15) can be rewritten in the following form:

$$\begin{pmatrix} \underset{\sim}{A}_s \underset{\sim}{G}_s \underset{\sim}{A}_s^T & \underset{\sim}{A}_t & \underset{\sim}{0} \\ \underset{\sim}{A}_t^T & -\underset{\sim}{Z}_t & \underset{\sim}{A}_{rt}^T \\ \underset{\sim}{0} & \underset{\sim}{A}_{rt} & \underset{\sim}{A}_{rr} \underset{\sim}{G}_r \underset{\sim}{A}_{rr}^T \end{pmatrix} \begin{pmatrix} \underset{\sim}{v}_s \\ \underset{\sim}{i}_t \\ \underset{\sim}{v}_r \end{pmatrix} = \begin{pmatrix} \underset{\sim}{J}_{ss} \\ \underset{\sim}{E}_{ts} \\ \underset{\sim}{J}_{rs} \end{pmatrix} \tag{5.16}$$

Let us now examine the term $\underset{\sim}{A}_s \underset{\sim}{G}_s \underset{\sim}{A}_s^T$ in Eq. (5.16). Substituting $\underset{\sim}{A}_s$ of Eq. (5.2) and $\underset{\sim}{G}_s$ of Eq. (5.3) into $\underset{\sim}{A}_s \underset{\sim}{G}_s \underset{\sim}{A}_s^T$, we obtain

$$\underset{\sim}{A}_s \underset{\sim}{G}_s \underset{\sim}{A}_s^T = \begin{array}{c} \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_k \end{array} \begin{array}{c} \begin{array}{cccc} \alpha_1 & \alpha_2 & \cdots & \alpha_k \end{array} \\ \begin{pmatrix} \underset{\sim}{Y}_{s1} & & & \\ & \underset{\sim}{Y}_{s2} & & \\ & & \ddots & \underset{\sim}{0} \\ & \underset{\sim}{0} & & \ddots \\ & & & & \underset{\sim}{Y}_{sk} \end{pmatrix} \end{array} \tag{5.17}$$

where $\underset{\sim}{Y}_{sj} = \underset{\sim}{A}_{sj} \underset{\sim}{G}_{sj} \underset{\sim}{A}_{sj}^T$ $j=1,2,\ldots\ldots,k$,

so Eq (5.16) can be rewritten as:

$$\left(\begin{array}{ccc:c:c} \underset{\sim}{Y} & & & \underset{\sim}{A}_{t1} & \\ & \underset{\sim}{Y}_{s2} & \underset{\sim}{0} & \underset{\sim}{A}_{t2} & \\ & \ddots & & \vdots & \underset{\sim}{0} \\ \underset{\sim}{0} & & & \vdots & \\ & & \underset{\sim}{Y}_{sk} & \underset{\sim}{A}_{tk} & \\ \hdashline \underset{\sim}{A}_{t1}^T & \underset{\sim}{A}_{t2}^T & \underset{\sim}{A}_{tk}^T & -\underset{\sim}{Z}_t & \underset{\sim}{A}_{rt}^T \\ \hdashline & \underset{\sim}{0} & & \underset{\sim}{A}_{rt} & \underset{\sim}{Y}_r \end{array}\right) \left(\begin{array}{c} \underset{\sim}{v}_{r1} \\ \underset{\sim}{v}_{r2} \\ \vdots \\ \underset{\sim}{v}_{rk} \\ \hdashline \underset{\sim}{i}_t \\ \hdashline \underset{\sim}{v}_r \end{array}\right) = \left(\begin{array}{c} \underset{\sim}{J}_{ss1} \\ \underset{\sim}{J}_{ss2} \\ \vdots \\ \underset{\sim}{J}_{ssk} \\ \hdashline \underset{\sim}{E}_{ts} \\ \hdashline \underset{\sim}{J}_{rs} \end{array}\right) \tag{5.18}$$

where $\underset{\sim}{Y}_r = \underset{\sim}{A}_{rr}\underset{\sim}{G}_r\underset{\sim}{A}_{rr}^T$. Eq. (5.18) is the resulting matrix by branch tearing method, which has the desirable bordered block-diagonal structure and it is a particular ordering of the hybrid analysis equations.

5.2. Derivation of the Node Tearing Method

Let the interconnection be defined as the remaining part of the network when all the subnetworks are removed, that is, the set of tearing nodes and the rest of network in Fig. 5.3. Let us assume that a proper set of tearing nodes has been chosen such that no coupling exists either among the torn subnetworks or between the torn subnetworks and the interconnection. The node set α is partitioned into three subsets α_s, α_t and α_r, and the branch set $\mathcal{B}$ is partitioned into two subsets $\mathcal{B}_s$ and $\mathcal{B}_r$ (Fig. 5.3). This yields the following special structures for the reduced incidence matrix $\underset{\sim}{A}$ and the branch conductance matrix $\underset{\sim}{G}$ of the network N:

$$\underset{\sim}{A} = \begin{array}{c} \\ \alpha_s \\ \alpha_t \\ \alpha_r \end{array} \begin{array}{c} \begin{array}{cc} \mathcal{B}_s & \mathcal{B}_r \end{array} \\ \begin{bmatrix} \underset{\sim}{A}_s & \underset{\sim}{0} \\ \underset{\sim}{A}_{ts} & \underset{\sim}{A}_{tr} \\ \underset{\sim}{0} & \underset{\sim}{A}_r \end{bmatrix} \end{array} \tag{5.19}$$

$$\underset{\sim}{G} = \begin{array}{c} \\ \mathcal{B}_s \\ \mathcal{B}_r \end{array} \begin{array}{c} \begin{array}{cc} \mathcal{B}_s & \mathcal{B}_r \end{array} \\ \begin{bmatrix} \underset{\sim}{G}_s & \underset{\sim}{0} \\ \underset{\sim}{0} & \underset{\sim}{G}_r \end{bmatrix} \end{array} \tag{5.20}$$

Let the torn subnetwork have k subnetworks $N_1, N_2, \ldots\ldots, N_k$. Let α_s and β_s be partitioned correspondingly into k subsets $\alpha_1, \alpha_2, \ldots\ldots, \alpha_k$ and $\beta_1, \beta_2, \ldots\ldots, \beta_k$ respectively. With this partitioning, the node-branch incidence matrix $\underset{\sim}{A}$ can be written as:

$$\underset{\sim}{A} = \begin{array}{c|cccc|c} & \beta_1 & \beta_2 & \ldots\ldots & \beta_k & \beta_r \\ \alpha_1 & \underset{\sim}{A}_{s1} & & & & \\ \alpha_2 & & \underset{\sim}{A}_{s2} & & \underset{\sim}{0} & \\ \vdots & & \ddots & & & \underset{\sim}{0} \\ & & \underset{\sim}{0} & & & \\ \alpha_k & & & & \underset{\sim}{A}_{sk} & \\ \hline \alpha_t & \underset{\sim}{A}_{ts1} & \underset{\sim}{A}_{ts2} & & \underset{\sim}{A}_{tsk} & \underset{\sim}{A}_{tr} \\ \hline \alpha_r & & \underset{\sim}{0} & & & \underset{\sim}{A}_{r} \end{array} \tag{5.21}$$

The branch conductance matrix $\underset{\sim}{G}$ can be written as:

$$\underset{\sim}{G} = \begin{array}{c|cccc|c} & \beta_1 & \beta_2 & \ldots & \beta_k & \beta_r \\ \beta_1 & \underset{\sim}{G}_{s1} & & & & \\ \beta_2 & & \underset{\sim}{G}_{s2} & & \underset{\sim}{0} & \\ \vdots & & \ddots & & & \underset{\sim}{0} \\ & & \underset{\sim}{0} & & & \\ \beta_k & & & & \underset{\sim}{G}_{sk} & \\ \hline \beta_r & & \underset{\sim}{0} & & & \underset{\sim}{G}_{r} \end{array} \tag{5.22}$$

The network variables are constrained by the Kirchhoff's current law (KCL), Kirchhoff's voltage law (KVL) and the branch constraint relations (BC).

$$\text{(KCL)} \quad \underset{\sim}{A}_{s}\underset{\sim}{i}_{s} = \underset{\sim}{0} \tag{5.23}$$

$$\underset{\sim}{A}_{ts}\underset{\sim}{i}_{s} + \underset{\sim}{A}_{tr}\underset{\sim}{i}_{r} = \underset{\sim}{0} \tag{5.24}$$

$$\underset{\sim}{A}_{r}\underset{\sim}{i}_{r} = \underset{\sim}{0} \tag{5.25}$$

$$\text{(KVL)} \quad \underset{\sim}{e}_{s} = \underset{\sim}{A}^{T}_{s}\underset{\sim}{v}_{s} + \underset{\sim}{A}^{T}_{ts}\underset{\sim}{v}_{t} \tag{5.26}$$

$$\underset{\sim}{e}_{r} = \underset{\sim}{A}^{T}_{tr}\underset{\sim}{v}_{t} + \underset{\sim}{A}^{T}_{r}\underset{\sim}{v}_{r} \tag{5.27}$$

$$\text{(BC)} \quad \underset{\sim}{i}_{s} = \underset{\sim}{I}_{ss} + \underset{\sim}{G}_{s}\underset{\sim}{e}_{s} - \underset{\sim}{G}_{s}\underset{\sim}{e}_{ss} \tag{5.28}$$

$$\underset{\sim}{i}_{r} = \underset{\sim}{I}_{rs} + \underset{\sim}{G}_{r}\underset{\sim}{e}_{r} - \underset{\sim}{G}_{r}\underset{\sim}{e}_{rs} \tag{5.29}$$

Substituting Eqs. (5.26) and (5.27) into Eqs. (5.28) and (5.29), and then substituting the results into Eqs. (5.23), (5.24) and (5.25), we obtain

$$\underset{\sim}{A}_{s}\underset{\sim}{G}_{s}\underset{\sim}{A}^{T}_{s}\underset{\sim}{v}_{s} + \underset{\sim}{A}_{s}\underset{\sim}{G}_{s}\underset{\sim}{A}^{T}_{ts}\underset{\sim}{v}_{t} = \underset{\sim}{J}_{ss} \tag{5.30}$$

$$\underset{\sim}{A}_{ts}\underset{\sim}{G}_{s}\underset{\sim}{A}^{T}_{s}\underset{\sim}{v}_{s} + (\underset{\sim}{A}_{ts}\underset{\sim}{G}_{s}\underset{\sim}{A}^{T}_{ts} + \underset{\sim}{A}_{tr}\underset{\sim}{G}_{r}\underset{\sim}{A}^{T}_{tr})\underset{\sim}{v}_{t} + \underset{\sim}{A}_{tr}\underset{\sim}{G}_{r}\underset{\sim}{A}^{T}_{r}\underset{\sim}{v}_{r} = \underset{\sim}{J}_{ts} \tag{5.31}$$

$$\underset{\sim}{A}_{r}\underset{\sim}{G}_{r}\underset{\sim}{A}^{T}_{tr}\underset{\sim}{v}_{t} + \underset{\sim}{A}_{r}\underset{\sim}{G}_{r}\underset{\sim}{A}^{T}_{r}\underset{\sim}{v}_{r} = \underset{\sim}{J}_{rs} \tag{5.32}$$

where $\underset{\sim}{J}_{ss} \triangleq \underset{\sim}{A}_{s}\underset{\sim}{G}_{s}\underset{\sim}{e}_{ss} - \underset{\sim}{A}_{s}\underset{\sim}{I}_{ss}$,

$\underset{\sim}{J}_{ts} \triangleq \underset{\sim}{A}_{ts}\underset{\sim}{G}_{s}\underset{\sim}{e}_{ss} - \underset{\sim}{A}_{ts}\underset{\sim}{I}_{ss} + \underset{\sim}{A}_{tr}\underset{\sim}{G}_{r}\underset{\sim}{e}_{rs} - \underset{\sim}{A}_{tr}\underset{\sim}{I}_{rs}$,

and $\underset{\sim}{J}_{rs} \triangleq \underset{\sim}{A}_{r}\underset{\sim}{G}_{r}\underset{\sim}{e}_{rs} - \underset{\sim}{A}_{r}\underset{\sim}{I}_{rs}$.

Eqs. (5.30), (5.31) and (5.32) can be rewritten as:

$$\begin{bmatrix} \underset{\sim}{A}_s \underset{\sim}{G}_s \underset{\sim}{A}_s^T & \underset{\sim}{A}_s \underset{\sim}{G}_s \underset{\sim}{A}_{ts}^T & \underset{\sim}{0} \\ \underset{\sim}{A}_{ts} \underset{\sim}{G}_s \underset{\sim}{A}_s^T & \underset{\sim}{A}_{ts} \underset{\sim}{G}_s \underset{\sim}{A}_{ts}^T + \underset{\sim}{A}_{tr} \underset{\sim}{G}_r \underset{\sim}{A}_{tr}^T & \underset{\sim}{A}_{tr} \underset{\sim}{G}_r \underset{\sim}{A}_r^T \\ \underset{\sim}{0} & \underset{\sim}{A}_r \underset{\sim}{G}_r \underset{\sim}{A}_{tr}^T & \underset{\sim}{A}_r \underset{\sim}{G}_r \underset{\sim}{A}_r^T \end{bmatrix} \begin{bmatrix} \underset{\sim}{v}_s \\ \underset{\sim}{v}_t \\ \underset{\sim}{v}_r \end{bmatrix} = \begin{bmatrix} \underset{\sim}{J}_{ss} \\ \underset{\sim}{J}_{ts} \\ \underset{\sim}{J}_{rs} \end{bmatrix} \qquad (5.33)$$

Let us now examine the term $\underset{\sim}{A}_s \underset{\sim}{G}_s \underset{\sim}{A}_s^T$ in Eq. (5.33). Substituting the $\underset{\sim}{A}_s$ of Eq. (5.21) and $\underset{\sim}{G}_s$ of Eq. (5.22) into $\underset{\sim}{A}_s \underset{\sim}{G}_s \underset{\sim}{A}_s^T$, we obtain

$$\underset{\sim}{A}_s \underset{\sim}{G}_s \underset{\sim}{A}_s^T = \begin{array}{c} \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_k \end{array} \begin{array}{c} \begin{array}{cccc} \alpha_1 & \alpha_2 & \cdots & \alpha_k \end{array} \\ \begin{bmatrix} \underset{\sim}{Y}_{s1} & & & \\ & \underset{\sim}{Y}_{s2} & & \\ & & \ddots & \underset{\sim}{0} \\ & \underset{\sim}{0} & & \\ & & & \underset{\sim}{Y}_{sk} \end{bmatrix} \end{array} \qquad (5.34)$$

where $\underset{\sim}{Y}_{sj} = \underset{\sim}{A}_{sj} \underset{\sim}{G}_{sj} \underset{\sim}{A}_{sj}^T$ $j=1,2,\ldots\ldots,k$

, so Eq. (5.33) can be rewritten as:

$$\left[\begin{array}{ccc:c:c} \underset{\sim}{Y}_{s1} & & & \underset{\sim}{Y}_{st1} & \\ & \underset{\sim}{Y}_{s2} & \underset{\sim}{0} & \underset{\sim}{Y}_{st2} & \\ & \ddots & & \vdots & \underset{\sim}{0} \\ \underset{\sim}{0} & & & \vdots & \\ & & \underset{\sim}{Y}_{sk} & \underset{\sim}{Y}_{stk} & \\ \hline \underset{\sim}{Y}_{ts1} & \underset{\sim}{Y}_{ts2} \cdots & \underset{\sim}{Y}_{tsk} & \underset{\sim}{Y}_{tt} & \underset{\sim}{Y}_{tr} \\ \hline & \underset{\sim}{0} & & \underset{\sim}{Y}_{rt} & \underset{\sim}{Y}_{rr} \end{array}\right] \left[\begin{array}{c} \underset{\sim}{v}_{s1} \\ \underset{\sim}{v}_{s2} \\ \vdots \\ \underset{\sim}{v}_{sk} \\ \hline \underset{\sim}{v}_t \\ \hline \underset{\sim}{v}_r \end{array}\right] = \left[\begin{array}{c} \underset{\sim}{J}_{ss1} \\ \underset{\sim}{J}_{ss2} \\ \vdots \\ \underset{\sim}{J}_{ssk} \\ \hline \underset{\sim}{J}_{ts} \\ \hline \underset{\sim}{J}_{rs} \end{array}\right] \qquad (5.35)$$

where $\underset{\sim}{Y}_{stj} = \underset{\sim}{A}_{sj}\underset{\sim}{G}_{sj}\underset{\sim}{A}^{T}_{ts}$ $j=1,2,\ldots\ldots,k$

$$,\underset{\sim}{Y}_{tsj} = \underset{\sim}{A}_{tsj}\underset{\sim}{G}_{sj}\underset{\sim}{A}^{T}_{sj} \quad j=1,2,\ldots\ldots,k$$

$$,\underset{\sim}{Y}_{tt} = \underset{\sim}{A}_{ts}\underset{\sim}{G}_{s}\underset{\sim}{A}^{T}_{ts} + \underset{\sim}{A}_{tr}\underset{\sim}{G}_{r}\underset{\sim}{A}^{T}_{tr}$$

$$,\underset{\sim}{Y}_{rt} = \underset{\sim}{A}_{tr}\underset{\sim}{G}_{r}\underset{\sim}{A}^{T}_{r}$$

$$,\underset{\sim}{Y}_{rt} = \underset{\sim}{A}_{r}\underset{\sim}{G}_{r}\underset{\sim}{A}^{T}_{tr}$$

and $\underset{\sim}{Y}_{rr} = \underset{\sim}{A}_{r}\underset{\sim}{G}_{r}\underset{\sim}{A}^{T}_{r}$.

Eq. (5.35) is the resulting matrix by node tearing method, which has the desirable bordered block-diagonal structure and is a particular ordering of the nodal analysis equations. In the above derivation the modified nodal method could have been used. In this case the vectors $\underset{\sim}{v}_{s}$ and $\underset{\sim}{v}_{r}$ consist of both node voltages and currents of branches for which an admittance description presents difficulties. This is actually the formulation used in the program SLATE.

5.3. Comparison of the Branch Tearing Method with the Node Tearing Method

As described above, branch tearing is equivalent to a particular ordering of the hybrid equations, node tearing is equivalent to a particular ordering of the nodal equations, so branch tearing requires the use of tearing branch currents as extra variables. As a result of the

above property, node tearing possesses the following advantages over branch tearing:

(1) the dimension of the matrix formulated by node tearing is smaller than that formulated by branch tearing;

(2) the number of nonzero entries in the matrix formulated by node tearing is smaller than that formulated by branch tearing [20];

(3) for passive networks, node tearing generates a diagonally-dominant matrix, so any application of the Gaussian elimination method with diagonal pivoting is stable, while this is not the the case in the branch tearing method;

(4) usually, in the analysis of large scale circuits, node tearing preserves the identities of the resulting torn subnetworks, while branch tearing sometimes destroys the identities of the torn subnetworks. However, one can generate examples in which the opposite is true, but these situations were not encountered in our examples.

The above conclusions are not conclusive; although the dimension of the matrix formulated by branch tearing is larger than that of node tearing, the extra nonzero entries are either +1 or -1. If this property is fully exploited, then node tearing may not be so advantageous. But full exploitation of the property that extra entries are either +1 or -1 requires a much more complicated sparse matrix technique. So node tearing is preferred and is used in the program SLATE.

5.4. Constructing the Node Tearing Matrix from Subnetworks

In Section 5.2, the derivation of node tearing is given; however, in the real implementation we do not want to solve the whole matrix equation at one time. We would like to process each subnetwork separately, and then obtain the solution of the entire network by combining the results of subnetwork process.

In the following the procedure of constructing the node tearing matrix from subnetworks is detailed. Let us consider one subnetwork N_i. Let the tearing nodes which are connected to N_i be denoted by α_{ti}, the node voltage of α_{ti} be denoted by $\underset{\sim}{v}_{ti}$, the nodes of N_i be denoted by α_i, the node voltages of α_i be denoted by $\underset{\sim}{v}_{si}$, and the currents which represents the relationship of the rest of the network with this subnetwork be denoted by $\underset{\sim}{J}_{ti}$ (Fig. 5.5). $\underset{\sim}{v}_{ti}$ and $\underset{\sim}{J}_{ti}$ satisfy the following relations:

$$\underset{\sim}{v}_t \triangleq \bigcup_i \underset{\sim}{v}_{ti} \tag{5.36}$$

$$\sum_i \underset{\sim}{J}_{ti} = \underset{\sim}{0} \qquad \text{(by KCL)} \tag{5.37}$$

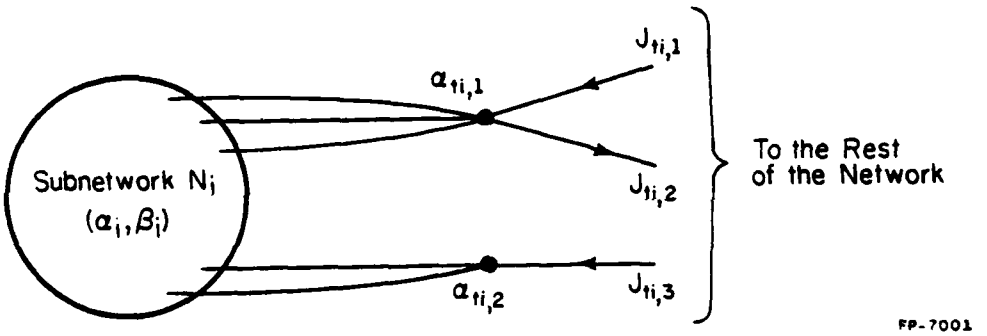


Fig. 5.5 One Subnetwork.

The node equations for this subnetwork have the following form:

$$\begin{array}{c} \\ \alpha_i \\ \alpha_{ti} \end{array} \begin{array}{c} \begin{array}{cc} \alpha_i & \alpha_{ti} \end{array} \\ \begin{bmatrix} \underset{\sim}{Y}_{si} & \underset{\sim}{Y}_{sti} \\ \underset{\sim}{Y}_{tsi} & \underset{\sim}{Y}_{tti} \end{bmatrix} \end{array} \begin{bmatrix} \underset{\sim}{v}_{si} \\ \underset{\sim}{v}_{ti} \end{bmatrix} = \begin{bmatrix} \underset{\sim}{J}_{ssi} \\ \underset{\sim}{J}_{tsi} \end{bmatrix} + \begin{bmatrix} \underset{\sim}{0} \\ \underset{\sim}{J}_{ti} \end{bmatrix} \tag{5.38}$$

By augmenting with appropriate zeros to match the dimension and summing Eq. (5.38) for all the subnetworks and the interconnection, we obtain

$$\left[\begin{array}{ccccc|c|c} \underset{\sim}{Y}_{s1} & & & & & \underset{\sim}{Y}_{st1} & \\ & \underset{\sim}{Y}_{s2} & & \underset{\sim}{0} & & \underset{\sim}{Y}_{st2} & \\ & & \ddots & & & \vdots & \underset{\sim}{0} \\ & \underset{\sim}{0} & & \ddots & & \vdots & \\ & & & & \underset{\sim}{Y}_{sk} & \underset{\sim}{Y}_{stk} & \\ \hline \underset{\sim}{Y}_{ts1} & \underset{\sim}{Y}_{ts2} & & & \underset{\sim}{Y}_{tsk} & \underset{\sim}{Y}_{tt} & \underset{\sim}{Y}_{tr} \\ \hline & & & \underset{\sim}{0} & & \underset{\sim}{Y}_{rt} & \underset{\sim}{Y}_{rr} \end{array}\right] \left[\begin{array}{c} \underset{\sim}{v}_{s1} \\ \underset{\sim}{v}_{s2} \\ \vdots \\ \underset{\sim}{v}_{sk} \\ \hline \underset{\sim}{v}_{t} \\ \hline \underset{\sim}{v}_{r} \end{array}\right] = \left[\begin{array}{c} \underset{\sim}{J}_{ss1} \\ \underset{\sim}{J}_{ss2} \\ \vdots \\ \underset{\sim}{J}_{ssk} \\ \hline \underset{\sim}{J}_{ts} \\ \hline \underset{\sim}{J}_{rs} \end{array}\right] \tag{5.39}$$

We can see that Eq. (5.39) is identical to Eq. (5.35).

Eq. (5.39) is solved by first eliminating all the $\underset{\sim}{Y}_{tsi}$ to obtain the interconnection matrix equations.

$$\begin{pmatrix} \underset{\sim}{Y}^{*}_{tt} & \underset{\sim}{Y}_{tr} \\ \underset{\sim}{Y}_{rt} & \underset{\sim}{Y}_{rr} \end{pmatrix} \begin{pmatrix} \underset{\sim}{v}_{t} \\ \underset{\sim}{v}_{r} \end{pmatrix} = \begin{pmatrix} \underset{\sim}{J}^{*}_{ts} \\ \underset{\sim}{J}_{rs} \end{pmatrix} \tag{5.40}$$

where $\underset{\sim}{Y}^{*}_{tt} = \underset{\sim}{Y}_{tt} - \sum_{i=1}^{k} \underset{\sim}{Y}_{tsi} \left(\underset{\sim}{Y}_{si}\right)^{-1} \underset{\sim}{Y}_{sti}$

and $\underset{\sim}{J}^{*}_{st} = \underset{\sim}{J}_{st} - \sum_{i=1}^{k} \underset{\sim}{Y}_{tsi} \left(\underset{\sim}{Y}_{si}\right)^{-1} \underset{\sim}{J}_{ssi}$

Eq. (5.40) is solved to obtain solutions for $\underset{\sim}{v}_{t}$ and $\underset{\sim}{v}_{r}$, and then the solution $\underset{\sim}{v}_{si}$ can be obtained by using backward substitution.

The above solution procedure can be modified to enable us to process each subnetwork separately to obtain the interconnection matrix equations. Let us consider Eq. (5.38) again, after eliminating $\underset{\sim}{Y}_{tsi}$, we have

$$\begin{array}{c} \\ \alpha_i \\ \\ \alpha_{ti} \end{array} \begin{array}{c} \begin{array}{cc} \alpha_i & \alpha_{ti} \end{array} \\ \left[\begin{array}{cc|c} & \underset{\sim}{U}_{si} & \underset{\sim}{L}^{-1}_{si}\ \underset{\sim}{Y}_{sti} \\ \underset{\sim}{L}_{si} & & \\ \hline \underset{\sim}{Y}_{tsi}\ \underset{\sim}{U}^{-1}_{si} & & \underset{\sim}{Y}^{*}_{tti} \end{array}\right] \end{array} \begin{pmatrix} \underset{\sim}{v}_{si} \\ \hline \underset{\sim}{v}_{ti} \end{pmatrix} = \begin{pmatrix} \underset{\sim}{L}^{-1}_{si}\ \underset{\sim}{J}_{ssi} \\ \hline \underset{\sim}{J}^{*}_{tsi} \end{pmatrix} + \begin{pmatrix} \underset{\sim}{0} \\ \hline \underset{\sim}{J}_{ti} \end{pmatrix} \tag{5.41}$$

where $\underset{\sim}{L}_{si}\underset{\sim}{U}_{si} = \underset{\sim}{Y}_{si}$,

$$\underset{\sim}{Y}^{*}_{tti} = \underset{\sim}{Y}_{tti} - \underset{\sim}{Y}_{tsi} \left(\underset{\sim}{Y}_{si}\right)^{-1} \underset{\sim}{Y}_{sti},$$

and $\underset{\sim}{J}^{*}_{tsi} = \underset{\sim}{J}_{tsi} - \underset{\sim}{Y}_{tsi} (\underset{\sim}{Y}_{si})^{-1} \underset{\sim}{J}_{ssi}$.

The partial interconnection matrix equations obtained from Eq. (5.41) are

$$\begin{bmatrix} \underset{\sim}{Y}^{*}_{tti} \end{bmatrix} \begin{bmatrix} \underset{\sim}{v}_{ti} \end{bmatrix} = \begin{bmatrix} \underset{\sim}{J}^{*}_{tsi} \end{bmatrix} + \begin{bmatrix} \underset{\sim}{J}_{ti} \end{bmatrix} \tag{5.42}$$

By augmenting the above arrays with appropriate zeros to match the dimension of $\underset{\sim}{Y}^{*}_{tt}$ (this is only conceptual, because sparse matrix techniques are used and every elment is put in the appropriate location in a one dimensional array) and summing up Eq. (5.42) for all the subnetworks and the interconnection, we obtain Eq. (5.40) again.

Since $\sum_i \underset{\sim}{J}_{ti} = \underset{\sim}{0}$ by KCL, therefore $\underset{\sim}{J}_{ti}$ does not appear in the final matrix equations (5.39) and (5.40), so we can neglect $\underset{\sim}{J}_{ti}$ in both Eqs. (5.38) and (5.42).

So the modified solution procedure is as follows:

(1) Formulate the simplified Eq. (5.38) for each subnetwork, i.e.

$$\begin{array}{c} \\ \alpha_i \\ \alpha_{ti} \end{array} \begin{array}{c} \begin{array}{cc} \alpha_i & \alpha_{ti} \end{array} \\ \begin{bmatrix} \underset{\sim}{Y}_{si} & \underset{\sim}{Y}_{sti} \\ \underset{\sim}{Y}_{tsi} & \underset{\sim}{Y}_{tti} \end{bmatrix} \end{array} \begin{bmatrix} \underset{\sim}{v}_{si} \\ \underset{\sim}{v}_{ti} \end{bmatrix} = \begin{bmatrix} \underset{\sim}{J}_{ssi} \\ \underset{\sim}{J}_{tsi} \end{bmatrix} \tag{5.43}$$

(2) Process Eq. (5.43) to obtain the simplified Eq. (5.42) for each subnetwork, i.e.

$$\begin{bmatrix} \underset{\sim}{Y}^{*}_{tti} \end{bmatrix} \begin{bmatrix} \underset{\sim}{v}_{ti} \end{bmatrix} = \begin{bmatrix} \underset{\sim}{J}^{*}_{tsi} \end{bmatrix} \tag{5.44}$$

(3) Sum up Eqs. (5.44) for all subnetworks and the interconnection with appropriate dimension match to obtain Eq. (5.40);

(4) Solve Eq. (5.40) to obtain $\underset{\sim}{V}_t$ and $\underset{\sim}{V}_r$;

(5) Solve the upper part of Eq. (5.41) by backward substitution to obtain $\underset{\sim}{V}_{si}$ for each subnetwork $\underset{\sim}{N}_i$.

5.5. Sparsity Considerations for the Node Tearing Method

Now let us compare different ways of sparsity exploitation for processing Eq. (5.43). For the solution procedure discussed in the previous section, both LU factorization and substitution procedures are required. In SLATE the modified nodal equation formulation and the new reordering strategy described in Chapter 2 are used at the subnetwork level. After the current variables and the corresponding 'positive' node voltage variables are eliminated, the final subnetwork matrix is structurally symmetric. So here we assume that the subnetwork matrix is structurally symmetric, under this assumption, there are two possible LU factorization procedures we would like to compare. there are other procedures described in [38], but these procedures are either equivalent to them or are less efficient.

The two LU factorization procedures are denoted by F_1 and $\bar{F}_2$ [38], and are given in Table 5.1.

where $\underset{\sim}{L}_{sk}\underset{\sim}{U}_{sk} = \underset{\sim}{Y}_{sk}$, $\quad \underset{\sim}{V} = \underset{\sim}{L}_{sk}^{-1}\underset{\sim}{Y}_{stk}$,

$\underset{\sim}{W}^T = \underset{\sim}{Y}_{tsk}\underset{\sim}{U}_{sk}^{-1}$, $\quad \hat{\underset{\sim}{V}} = \underset{\sim}{U}_{sk}^{-1}\underset{\sim}{V}$,

and $\underset{\sim}{L}_{tk}\underset{\sim}{U}_{tk} = \underset{\sim}{Y}^*_{ttk} = \underset{\sim}{Y}_{ttk} - \underset{\sim}{Y}_{tsk}\underset{\sim}{U}_{sk}^{-1}\underset{\sim}{L}_{sk}^{-1}\underset{\sim}{Y}_{stk}$.

Table 5.1 Possible Factorization Procedures.

F_1	$\bar{F}_2$
$\begin{bmatrix} \underset{\sim}{L}_{sk} & \underset{\sim}{0} \\ \underset{\sim}{W}^T & \underset{\sim}{L}_{tk} \end{bmatrix} \begin{bmatrix} \underset{\sim}{U}_{sk} & \underset{\sim}{V} \\ \underset{\sim}{0} & \underset{\sim}{U}_{tk} \end{bmatrix}$	$\begin{bmatrix} \underset{\sim}{L}_{sk}\underset{\sim}{U}_{sk} & \underset{\sim}{0} \\ \underset{\sim}{Y}_{tsk} & \underset{\sim}{L}_{tk} \end{bmatrix} \begin{bmatrix} \underset{\sim}{I} & \hat{\underset{\sim}{V}} \\ \underset{\sim}{0} & \underset{\sim}{U}_{tk} \end{bmatrix}$

In $\bar{F}_2$, only those rows of $\hat{\underset{\sim}{V}}$ ($\hat{\underset{\sim}{V}}_{req}$) which are required to compute $\hat{\underset{\sim}{V}}_p$ are computed, $\hat{\underset{\sim}{V}}_p$ consists of those rows of $\hat{\underset{\sim}{V}}$ corresponding to the nonzero columns of $\underset{\sim}{Y}_{tsk}$.

Let $|.|$ denote the number of nonzero elements in a vector or a matrix, and M(.j) and M(j.) the jth column and the jth row of matrix M, respectively. Let n_t denote the number of tearing nodes, and B(k) satisfies the following relation.

$$B(k) = \begin{cases} 0 & k = 0 \\ 1 & k \geq 1,\ k \text{ integer} \end{cases} \tag{5.45}$$

The following lemmas are used to compare the number of operations between F_1 and $\bar{F}_2$.

Lemma 5.1. Suppose the subnetwork matrix is structurally symmetric, then the number of rows of $\hat{\underset{\sim}{V}}_{req}$ equals the number of nonzero rows of $\underset{\sim}{V}$.

Proof: From the definition of $\hat{\underset{\sim}{V}}_p$, we know that any nonzero row of $\underset{\sim}{V}$ which is not a row of $\hat{\underset{\sim}{V}}_p$ must consist of only fill-ins. Suppose it is row i, then there must be a nonzero row j of $\hat{\underset{\sim}{V}}_p$, j < i, and a nonzero $\underset{\sim}{L}_{sk,ij}$. Row j together with $\underset{\sim}{L}_{sk,ij}$ creates the fill-ins of row i. Due to structure symmetry, there also must be a nonzero $\underset{\sim}{U}_{sk,ji}$ So in order to evaluate the row j of $\hat{\underset{\sim}{V}}_p$, it is necessary to evaluate the row i of $\hat{\underset{\sim}{V}}$.

Lemma 5.2. Suppose the subnetwork matrix is structurally symmetric and $\underset{\sim}{Y}_{sk}$ is mxm, then the difference in the number of operations between F_1 and $\bar{F}_2$ (DNF) is:

$$DNF = \sum_{j=1}^{m} |\underset{\sim}{V}(j\cdot)||\underset{\sim}{V}(j\cdot)| + \sum_{j=1}^{m} (|\underset{\sim}{V}_{sk}(j\cdot)| - 1)|\underset{\sim}{V}(j\cdot)|$$

$$- \sum_{j=2}^{m} \sum_{i=j+1}^{m} |\underset{\sim}{U}_{sk}(ji)||\hat{\underset{\sim}{V}}(i\cdot)|B(|\underset{\sim}{V}(i\cdot)| - \sum_{j=1}^{m} |\underset{\sim}{Y}_{tsk}(\cdot j)||\hat{\underset{\sim}{V}}(j\cdot)|B(|\underset{\sim}{V}(j\cdot)|) \quad (5.46)$$

If $\hat{\underset{\sim}{V}}_{req}$ is full, then

$$DNF = \sum_{j=1}^{m} |\underset{\sim}{V}(j\cdot)||\underset{\sim}{V}(j\cdot)| - \sum_{j=1}^{m} (|\underset{\sim}{U}_{sk}(j\cdot)| - 1)(n_t - |\underset{\sim}{V}(j\cdot)|)B(|\underset{\sim}{V}(j\cdot)|)$$

$$- n_t|\underset{\sim}{Y}_{tsk}| \quad (5.47)$$

Proof: $$DNF = \sum_{j=1}^{m} (|\underset{\sim}{U}_{sk}(\cdot j)| - 1)|\underset{\sim}{W}(j\cdot)| + \sum_{j=1}^{m} |\underset{\sim}{W}^T(\cdot j)||\underset{\sim}{V}(j\cdot)|$$

$$- \sum_{j=2}^{m} \sum_{i=j+1}^{m} |\underset{\sim}{U}_{sk}(ji)||\hat{\underset{\sim}{V}}_{req}(i\cdot)| - \sum_{j=1}^{m} |\underset{\sim}{Y}_{tsk}(\cdot j)||\hat{\underset{\sim}{V}}_{req}(j\cdot)| \quad (5.48)$$

From structure symmetry, we obtain

$$|\underset{\sim}{W}(j\cdot)| = |\underset{\sim}{V}(j\cdot)| \quad (5.49)$$

$$|\underset{\sim}{W}^T(\cdot j)| = |\underset{\sim}{V}(j\cdot)| \quad (5.50)$$

$$|\underset{\sim}{U}^T_{sk}(\cdot j)| = |\underset{\sim}{U}_{sk}(j\cdot)| \quad (5.51)$$

From Lemma 5.1, we obtain

$$|\hat{\underset{\sim}{V}}_{req}(j\cdot)| = |\hat{\underset{\sim}{V}}(j\cdot)|B(|\underset{\sim}{V}(j\cdot)|) \quad (5.52)$$

Substituting Eqs. (5.49), (5.50), (5.51), and (5.52) into Eq. (5.48), we obtain Eq. (5.46).

If $\hat{\underset{\sim}{V}}_{req}$ is full, then

$$|\hat{\underset{\sim}{V}}(j\cdot)| = n_t \quad (5.53)$$

Substituting Eq. (5.53) into Eq. (5.46), we obtain Eq. (5.47).

Associated with F_1 and $\bar{F}_2$, there are five possible substitution procedures denoted by S_1, S_1^*, S_1^{**}, S_2^*, and S_2^{**} [38] which are given in Table 5.2, where $\underset{\sim}{b}_{req}$ consists of the rows of $\underset{\sim}{b}$ which are required to compute $\underset{\sim}{b}_p$. $\underset{\sim}{b}_p$ are those rows of $\underset{\sim}{b}$ corresponding to the nonzero columns of $\underset{\sim}{Y}_{tsk}$.

Let C(.) denote the number of operations required in performing a given procedure S. By comparing the entries of the five substitution procedures in Table 5.2, the following equation is obtained.

$$C(S_1^*) - C(S_1^{**}) = C(S_2^*) - C(S_2^{**}) \tag{5.54}$$

In large scale integrated circuits, most of devices are nonlinear. Due to the current sources generated by Newton-Raphson iteration and numerical integration, it is reasonable to assume that the source vectors $\underset{\sim}{J}_{ssk}$ and $\underset{\sim}{J}_{tsk}$ are full. Also, since $\underset{\sim}{V}_{sk}$ and $\underset{\sim}{V}_{tk}$ are the required node voltages, it is reasonable to assume that they are full. Under the above assumptions, we obtain the following lemmas.

Lemma 5.3. $\underset{\sim}{a}$, $\underset{\sim}{b}$, $\underset{\sim}{y}$, and $\hat{\underset{\sim}{a}}$ are full.

Lemma 5.4. Suppose that the subnetwork matrix is structurally symmetric, then the number of rows of $\underset{\sim}{b}_{req}$ equals the number of nonzero rows of $\underset{\sim}{V}$.

Lemma 5.5. Suppose that that subnetwork matrix is structurally symmetric and $\underset{\sim}{Y}_{sk}$ is mxm, then the difference in the number of operations between S_1 and S_1^* (DNS1) is:

Table 5.2 Possible Substitution Procedures.

S_1	S_1^*	S_1^{**}	S_2^*	S_2^{**}
$L_{sk} a = J_{ssk}$ $y = W^T a$	$L_{sk} a = J_{ssk}$ $U_{sk} b_{req} = a$ $y = Y_{tsk} b_p$	$L_{sk} a = J_{ssk}$ $U_{sk} b_{req} = a$ $y = Y_{tsk} b_p$	$L_{sk} a = J_{ssk}$ $U_{sk} b_{req} = a$ $y = Y_{tsk} b_p$	$L_{sk} a = J_{ssk}$
$L_{tk} U_{tk} v_{tk} = J_{tsk} - y$	$L_{tk} U_{tk} v_{tk} = J_{tsk} - y$	$L_{tk} U_{tk} v_{tk} = J_{tsk} - y$	$L_{tk} U_{tk} = J_{tsk} - y$	$L_{tk} U_{tk} v_{tk} = J_{tsk} - y$
$Z_1 = V v_{tk}$ $\hat{a} = a - Z_1$ $U_{sk} v_{sk} = \hat{a}$	$Z_1 = V v_{tk}$ $\hat{a} = a - Z_1$ $U_{sk} v_{sk} = \hat{a}$	$Z = Y_{stk} v_{tk}$ $L_{sk} Z_1 = Z$ $\hat{a} = a - Z_1$ $V_{sk} v_{sk} = \hat{a}$	$Z_1 = V v_{tk}$ $U_{sk} Z_2 = Z_1$ $v_{sk} = b - Z_2$	$Z = Y_{stk} v_{tk}$ $L_{sk} Z_1 = Z$ $U_{sk} Z_2 = Z_1$ $v_{sk} = b - Z_2$

$$DNS1 = C(S_1) - C(S_1^*)$$

$$= \sum_{j=1}^{m} |\underset{\sim}{V}(j\cdot)| - \sum_{j=1}^{m} (|\underset{\sim}{U}_{sk}(j\cdot)| - 1)B(|\underset{\sim}{V}(j\cdot)|) - \sum_{j=1}^{m} |\underset{\sim}{Y}_{tsk}(\cdot j)|$$

$$= |\underset{\sim}{V}| - |\underset{\sim}{Y}_{tsk}| - \sum_{j=1}^{m} (|\underset{\sim}{U}_{sk}(j\cdot)| - 1)B(|\underset{\sim}{V}(j\cdot)|)$$

$$= (\text{number of fill-ins in } \underset{\sim}{V}) - \sum_{j=1}^{m} (|\underset{\sim}{U}_{sk}(j\cdot)| - 1)B(|\underset{\sim}{V}(j\cdot)|) \qquad (5.55)$$

Lemma 5.6. Suppose that the subnetwork matrix is structurally symmetric and $\underset{\sim}{Y}_{sk}$ is mxm, then the difference of number of operations between S_1 and S_1^{**} (DNS2) is:

$$DNS2 = C(S_1) - C(S_1^{**})$$

$$= DNS1 + \sum_{j=1}^{n_t} |\underset{\sim}{V}(\cdot j)| - \sum_{j=1}^{n_t} |\underset{\sim}{Y}_{stk}(\cdot j)| - \sum_{j=1}^{m} |\underset{\sim}{L}_{sk}(\cdot j)|B(|\underset{\sim}{V}(j\cdot)|)$$

$$= DNS1 + |\underset{\sim}{V}| - |\underset{\sim}{Y}_{tsk}| - \sum_{j=1}^{m} (|\underset{\sim}{U}_{sk}(j\cdot)| - 1)B(|\underset{\sim}{V}(j\cdot)|) - \sum_{j=1}^{m} B(|\underset{\sim}{V}(j\cdot)|)$$

$$= 2*DNS1 - \sum_{j=1}^{m} B(|\underset{\sim}{V}(j\cdot)|) \qquad (5.56)$$

Remark. From Lemma 5.6 we can conclude that if $C(S_1) \leq C(S_1^*)$ then $C(S_1) \leq C(S_1^{**})$, that is, only when $C(S_1) > C(S_1^*)$ do we need to compare S_1 with S_1^{**}.

Lemma 5.7. Suppose that the subnetwork matrix is structurally symmetric and $\underset{\sim}{Y}_{sk}$ is mxm, then

$$C(S_1^*) \leq C(S_2^*) \qquad (5.57)$$

$$C(S_1^{**}) \leq C(S_2^{**}) \qquad (5.58)$$

Proof: From Eq. (5.54), we obtain

$$C(S_1^*) - C(S_2^*) = C(S_1^{**}) - C(S_2^{**}) \tag{5.59}$$

So we only need to prove $C(S_1^*) - C(S_2^*) < 0$

$$\begin{aligned} C(S_1^*) - C(S_2^*) &= \sum_{j=1}^{m}(|\underset{\sim}{U}_{sk}(j\cdot)|-1)B(|\underset{\sim}{V}(j\cdot)|)-\sum_{j=1}^{m}(|\underset{\sim}{U}_{sk}(j\cdot)|-1) \\ &\quad +\sum_{j=1}^{m}(|\underset{\sim}{U}_{sk}(j\cdot)|-1)-\sum_{j=1}^{m}(|\underset{\sim}{U}_{sk}(j\cdot)|-1)|\underset{\sim}{Z}_2(j)| \\ &= \sum_{j=1}^{m}(|\underset{\sim}{U}_{sk}(j\cdot)|-1)(B(|\underset{\sim}{V}(j\cdot)1)-|\underset{\sim}{Z}_2(j)|) \end{aligned} \tag{5.60}$$

Since

$$|\underset{\sim}{Z}_1(j)| = B(|\underset{\sim}{V}(j\cdot)|) \tag{5.61}$$

and

$$|\underset{\sim}{Z}_2(j)| \geq |\underset{\sim}{Z}_1(j)| \tag{5.62}$$

so we have

$$|\underset{\sim}{Z}_2(j)| \geq B(|\underset{\sim}{V}(j\cdot)|) \tag{5.63}$$

Substituting Eq. (5.63) into Eq. (5.60), we obtain

$$C(S_1^*) - C(S_2^*) \leq 0 \tag{5.64}$$

From Lemma 5.7, we know that S_2 and S_2^{**} are not as efficient as the other procedures, so they are eliminated from the list of possible substitution procedures. Now we are left with S_1, S_1^* and S_1^{**}. The possible combinations of factorization methods and substitution methods are $F_1 + S_1$, $F_1 + S_1^*$, $F_1 + S_1^{**}$, $\bar{F}_2 + S_1^*$, and $\bar{F}_2 + S_1^{**}$. Theoretically we can not eliminate any of these five combinations, because we can always come up with a special subcircuit structure for which a particular combination gives the best result. However, after conducting a large number of studies, we found experimentally that $F_1 + S_1$ gives the best results for all the practical circuits we used; moreover, $F_1 + S_1$ is well compatible

with the sparse matrix techniques and is the easiest one to implement, so $F_1 + S_1$ was chosen to be used in the program SLATE.

In the following we would like to present a small selection of the examples which we have analyzed by using Lemma 5.2, Lemma 5.5 and Lemma 5.6.

Example 5.1. The subcircuit used is a TTL two-input NAND gate with parasitic resistors included (Fig. 5.6). The admittance matrix for this subcircuit is shown in Fig. 5.7.

DNF = 20 - 23 - 39 = -42

DNS1 = 9 - 13 = -4

DNS2 = -8 - 10 = -18

so the best combination for this subcircuit is $F_1 + S_1$.

Example 5.2. The subcircuit used is an ECL two-input NOR gate with parasitic resistors included (Fig. 5.8). The admittance matrix for this subcircuit is shown in Fig. 5.9.

DNF = 17 - 15 - 27 = -25

DNS1 = 6 - 8 = -2

DNS2 = -4 - 6 = -10

so the best combination for this subcircuit is $F_1 + S_1$.

Example 5.3. The subcircuit used is an MOS two-input NAND gate with

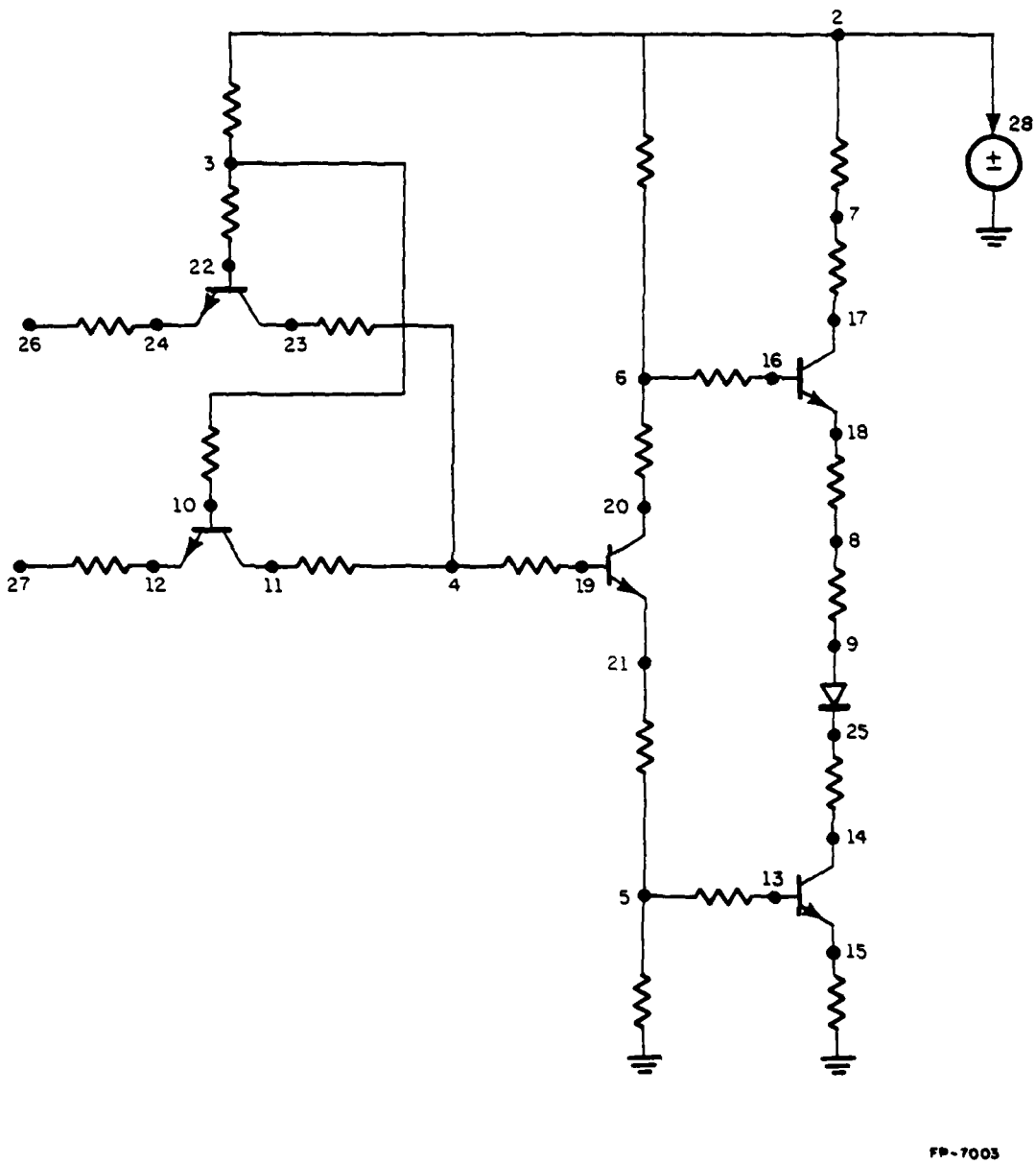


Fig. 5.6 TTL Two-Input NAND Gate.

		28	2	7	5	6	8	9	15	13	14	17	16	18	3	10	12	11	19	20	21	23	22	24	4	25	26	27
(2)	26	X	X	X	0	X	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0
(28)	2	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	7	0	X	X	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	5	0	0	0	X	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0
	6	0	X	0	0	X	0	0	0	0	0	0	X	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0
	8	0	0	0	0	0	X	X	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	9	0	0	0	0	0	X	X	0	0	0	0	0	F	0	0	0	0	0	0	0	0	0	0	0	X	0	0
	15	0	0	0	0	0	0	0	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	13	0	0	0	X	0	0	0	X	X	X	0	0	0	0	0	0	0	0	0	F	0	0	0	0	0	0	0
	14	0	0	0	0	0	0	0	X	X	X	0	0	0	0	0	0	0	0	0	F	0	0	0	0	X	0	0
	17	0	0	X	0	0	0	0	0	0	0	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	0	0	0	0	X	0	0	0	0	0	X	X	X	0	0	0	0	0	F	0	0	0	0	0	0	0	0
	18	0	0	0	0	0	X	F	0	0	0	X	X	X	0	0	0	0	0	F	0	0	0	0	0	F	0	0
	3	0	X	0	0	0	0	0	0	0	0	0	0	0	X	X	0	0	0	0	0	0	X	0	0	0	0	0
	10	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	0	0	0	0	F	0	0	0	0	0
	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	0	0	0	0	F	0	0	0	0	X
	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	0	0	0	0	F	0	X	0	0	F
	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	0	0	0	X	0	0	0
	20	0	0	0	0	X	0	0	0	0	0	0	F	F	0	0	0	0	X	X	X	0	0	0	F	F	0	0
	21	0	0	0	X	0	0	0	0	F	F	0	0	0	0	0	0	0	X	X	X	0	0	0	F	F	0	0
	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	0	0	0
	22	0	0	0	0	0	0	0	0	0	0	0	0	0	X	F	F	F	0	0	0	X	X	X	F	0	0	F
	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	F	0	X	F
	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	F	F	X	F	F	X	F	F	F
	25	0	0	0	0	0	0	X	0	0	X	0	0	F	0	0	0	0	0	F	F	0	0	0	F	X	F	F
	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	F	F	X	F
	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	F	0	0	0	0	F	F	F	F	F	X

Fig. 5.7 Admittance Matrix for the Subcircuit in Fig. 5.6.

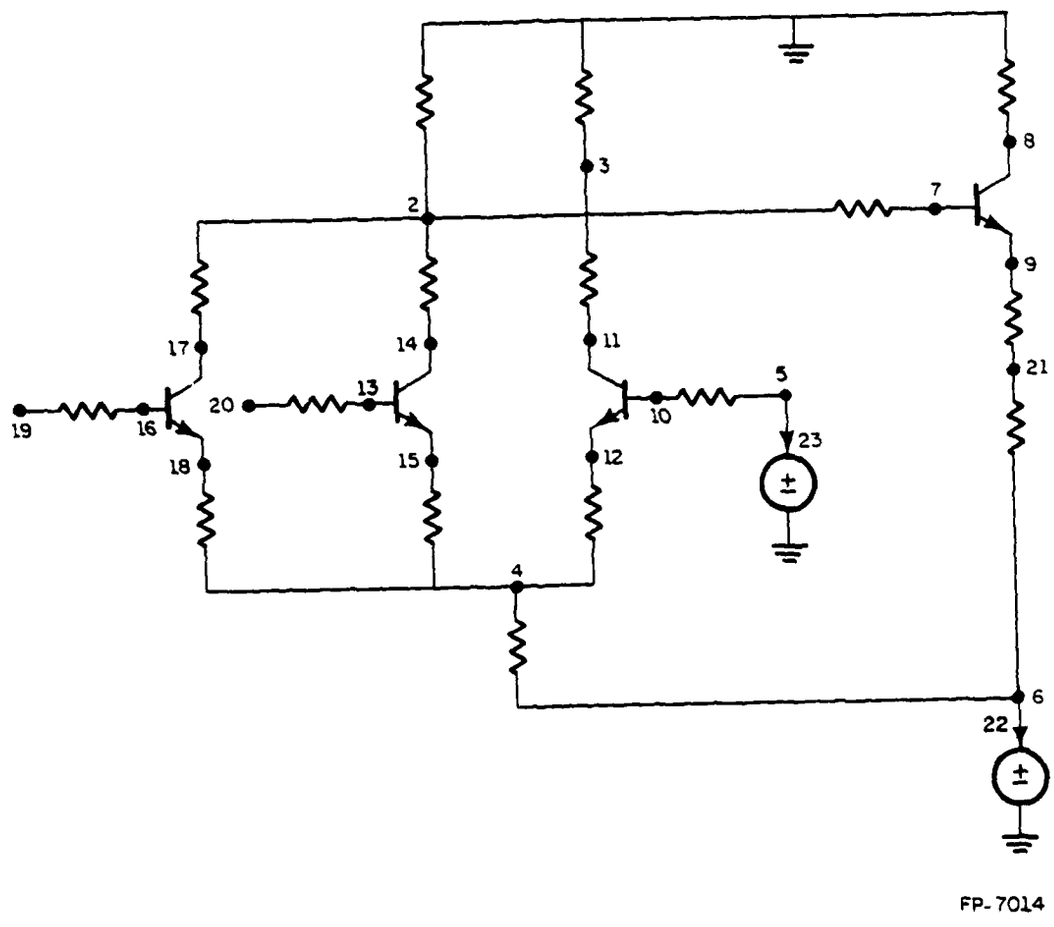


Fig. 5.8 ECL Two-Input NOR Gate.

		23	22	5	6	3	8	7	9	10	11	12	4	14	15	16	17	18	13	2	19	20	21
(5)	23	1	0	X	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0
(6)	22	0	1	0	X	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	X
(23)	5	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(22)	6	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	3	0	0	0	0	X	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0
	8	0	0	0	0	0	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	X	X	X	0	0	0	0	0	0	0	0	0	0	X	0	0	0
	9	0	0	0	0	0	X	X	X	0	0	0	0	0	0	0	0	0	0	F	0	0	X
	10	0	0	X	0	0	0	0	0	X	X	X	0	0	0	0	0	0	0	0	0	0	0
	11	0	0	0	0	X	0	0	0	X	X	X	0	0	0	0	0	0	0	0	0	0	0
	12	0	0	0	0	0	0	0	0	X	X	X	X	0	0	0	0	0	0	0	0	0	0
	4	0	0	0	X	0	0	0	0	0	0	X	X	0	X	0	0	X	0	0	0	0	0
	14	0	0	0	0	0	0	0	0	0	0	0	0	X	X	0	0	0	X	X	0	0	0
	15	0	0	0	0	0	0	0	0	0	0	0	X	X	X	0	0	F	X	F	0	0	0
	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	0	0	X	0	0
	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	0	X	F	0	0
	18	0	0	0	0	0	0	0	0	0	0	0	X	0	F	X	X	X	F	F	F	0	0
	13	0	0	0	0	0	0	0	0	0	0	0	0	X	X	0	0	F	X	F	F	X	0
	2	0	0	0	0	0	0	X	F	0	0	0	0	X	F	0	X	F	F	X	F	F	F
	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	F	F	F	F	X	F	F
	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	F	F	X	F
	21	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	F	F	F	X

Fig. 5.9 Admittance Matrix for the Subcircuit in Fig. 5.8.

parasitic resistors included (Fig. 5.10). The admittance matrix for this subcircuit is shown in Fig. 5.11.

DNF = 18 - 9 - 30 = -21

DNS1 = 3 - 5 = -2

DNS2 = -4 - 7 = -11

so the best combination for this subcircuit is $F_1 + S_1$.

These three examples show that at the subnetwork level the best combination is $F_1 + S_1$.

For the interconnection matrix, because this matrix has the same property as that of the matrix formulated by the MNA for any circuit, so the new reordering strategy of the MNA and the Markowitz sparse matrix scheme are used to exploit the sparsity at this level.

5.6. Implementation of the Node Tearing Method

As concluded in the previous section the best combination at the subcircuit level is in most cases $F_1 + S_1$, now we would like to describe the implementation of this approach. First, at the subcircuit level, the source vector is appended to the matrix to form

$$\begin{bmatrix} \underset{\sim}{Y}_{sk} & \underset{\sim}{Y}_{stk} & \underset{\sim}{J}_{ssk} \\ \underset{\sim}{Y}_{tsk} & \underset{\sim}{Y}_{ttk} & \underset{\sim}{J}_{tsk} \end{bmatrix} \tag{5.65}$$

Secondly, use the new reordering strategy of the MNA and the Markowitz sparse matrix scheme to find the LU factorization of $\underset{\sim}{Y}_{sk}$. The LU

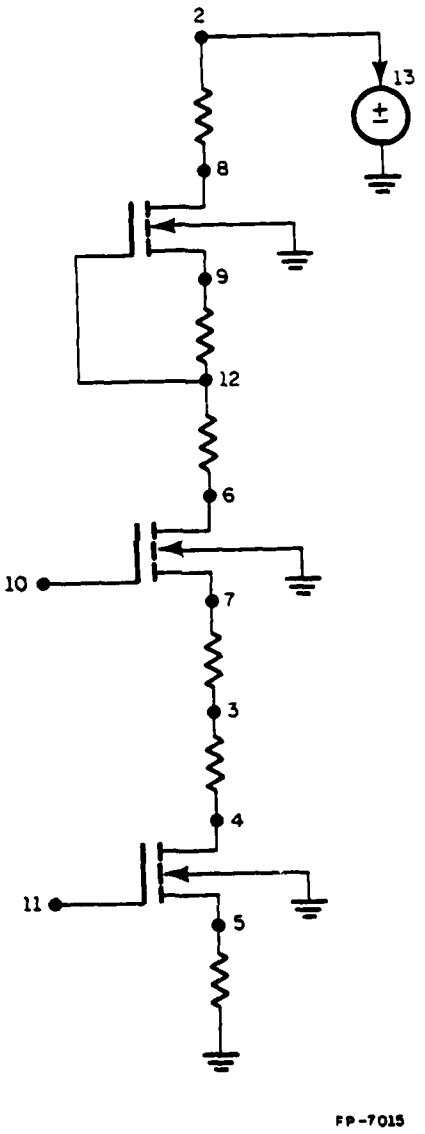


Fig. 5.10 MOS Two-Input NAND Gate.

		13	2	5	4	8	9	3	7	6	10	11	12
(2)	13	1	X	0	0	X	0	0	0	0	0	0	0
(13)	2	0	1	0	0	0	0	0	0	0	0	0	0
	5	0	0	X	X	0	0	0	0	0	0	X	0
	4	0	0	X	X	0	0	X	0	0	0	X	0
	8	0	X	0	0	X	X	0	0	0	0	0	X
	9	0	0	0	0	X	X	0	0	0	0	0	X
	3	0	0	0	X	0	0	X	X	0	0	F	0
	7	0	0	0	0	0	0	X	X	X	X	F	0
	6	0	0	0	0	0	0	0	X	X	X	F	X
	10	0	0	0	0	0	0	0	X	X	X	F	F
	11	0	0	X	X	0	0	F	F	F	F	X	F
	12	0	0	0	0	X	X	0	0	X	F	F	X

Fig. 5.11 Admittance Matrix for the Subcircuit in Fig. 5.10.

factorization operates on the whole matrix but terminates after the LU factorization of $\underset{\sim}{Y}_{sk}$ is obtained. Now the original matrix is transformed into

$$\begin{bmatrix} \underset{\sim}{L}_{sk} & \underset{\sim}{U}_{sk} & \underset{\sim}{L}_{sk}^{-1}\underset{\sim}{Y}_{stk} & \underset{\sim}{L}_{sk}^{-1}\underset{\sim}{J}_{ssk} \\ \underset{\sim}{Y}_{tsk}\underset{\sim}{U}_{sk}^{-1} & & \underset{\sim}{Y}^{*}_{ttk} & \underset{\sim}{J}^{*}_{tsk} \end{bmatrix} \tag{5.66}$$

Thirdly, formulate the interconnection matrix equation (5.40). Fourthly, use the new reordering strategy of the MNA and the Markowitz sparse matrix scheme to find the LU factorization of Eq. (5.40), and use forward and backward substitution to find the solutions for $\underset{\sim}{v}_t$ and $\underset{\sim}{v}_r$. Fifthly, use backward substitution to solve the upper part of Eq. (5.46) to obtain the solutions $\underset{\sim}{V}_{sk}$.

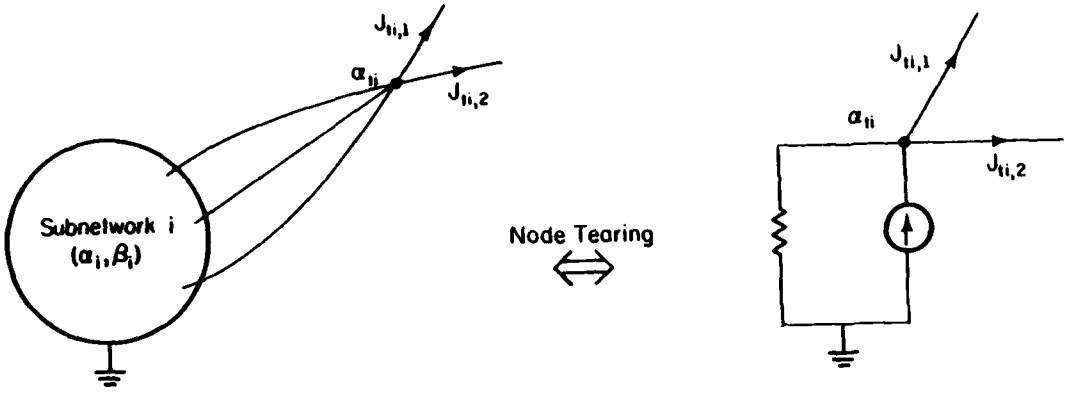
Remark: The reordering of the subnetwork and network matrix equations is done in the preprocessing phase. The subnetwork matrix equations are reordered first. So when the network matrix equations are reordered, the structures of all the $\underset{\sim}{Y}^{*}_{ttk}$'s are known. From the structures of all the $\underset{\sim}{Y}^{*}_{ttk}$'s and the circuit description of the network, we can reorder the network matrix equations by the new reordering strategy of the MNA and the Markowitz sparse matrix scheme.

5.7. Circuit Interpretation of the Tearing Methods

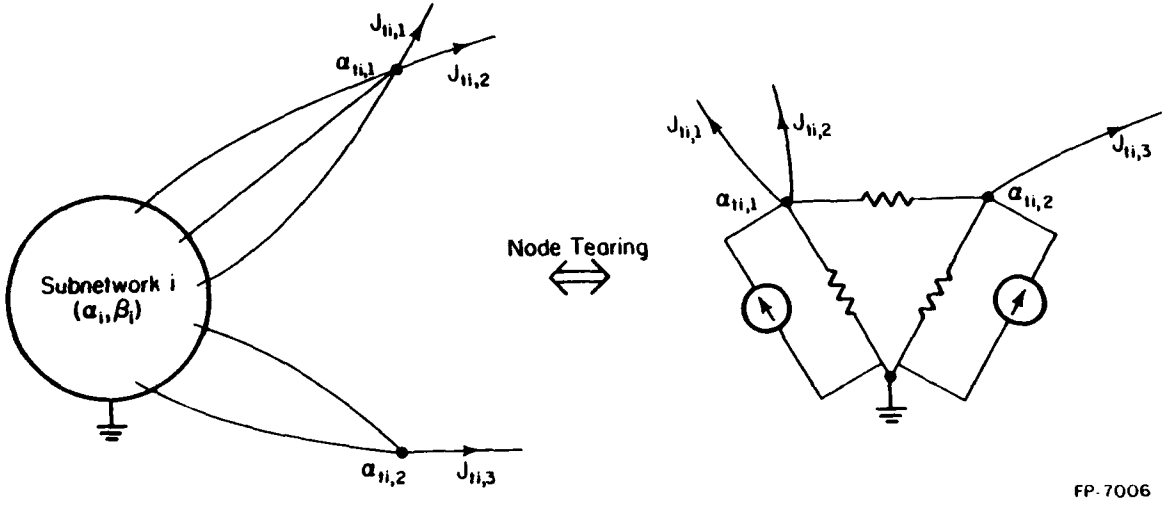
Although the derivations of the tearing methods are quite mathematical, there is a very simple circuit interpretation. The node tearing method is just a generalized Norton equivalent circuit approach. The branch tearing method is just a generalized Thevenin eqivalent circuit approach.

Let us consider node tearing first. Consider the special case when there is only one tearing node. The partial interconnection matrix equations (Eq. (5.42)) that we obtain for each subnetwork are just the Norton equivalent circuit matrix equations for that subnetwork. This is illustrated in Fig. 5.12(a). So for the case when there are more than one tearing node, Eq. (5.42) is just the generalized Norton equivalent circuit matrix equations for each subnec. See Fig. 5.12(b) for an illustration of the case of two tearing nodes.

Now let us consider branch tearing for the special case when there is only one tearing branch. The partial interconnection matrix equations that we obtain for each subnetwork are just the Thevenin equivalent circuit matrix equations for that subnetwork. This is illustrated in Fig. 5. 13(a). So for the case when there is more than one tearing branche, the partial interconnection matrix equations that we obtain for each subnetwork are just the generalized Thevenin eqivalent circuit matrix equations for that subnetwork Fig. 5.13(b) illustrates the case of two tearing branches.

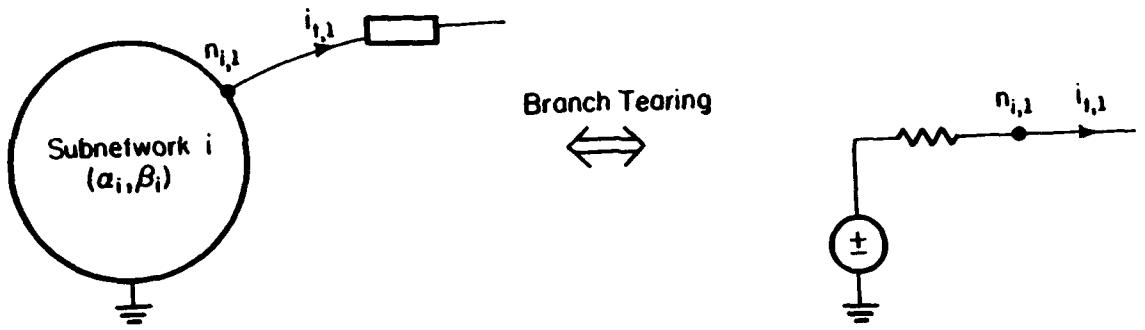


(a) One Tearing Node.

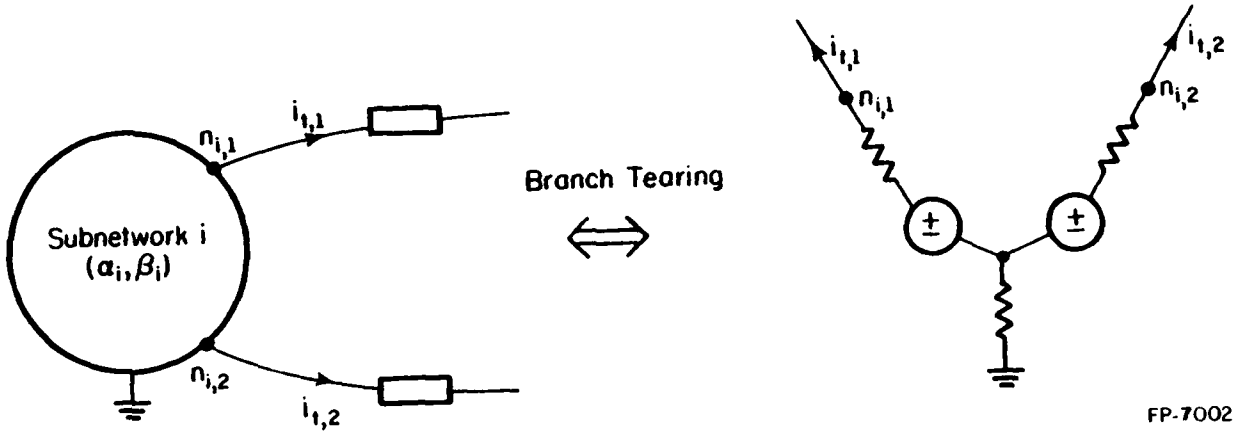


(b) Two Tearing Nodes.

Fig. 5.12 Norton Equivalent Circuit Interpretation of the Node Tearing Method.



(a) One Tearing Branch.



(b) Two Tearing Branch.

Fig. 5.13 Thevenin Equivalent Circuit Interpretation of the Branch Tearing Method.

5.8. Discussion and Conclusion

The reason why we considered many possible LU factorization and substitution procedures at the subnetwork level is that at the subnetwork level we only want to perform Gaussian elimination for the variables $\underset{\sim}{v}_{sk}$ and the elimination procedure terminates after the LU factorization of $\underset{\sim}{Y}_{sk}$ is obtained. At the interconnection level, because we perform the Gaussian elimination for all the variables, only one LU factorization and substitution procedure is used.

As discussed in Section 5.5, it is possible to generate special subcircuit structures for which some other combinations give better results, however, our experimental results show that even for these specially constructed circuits the difference of the number of operations is only 1 or 2 most of the time, so this very small savings does not justify the extra difficulties of implementation associated with these other combinations; moreover, for all the practical circuits we tested, $F_1 + S_1$ showed considerable savings over all the other combinations.

VI. LATENCY EXPLOITATION

In conventional circuit simulation programs [1,2], all of the node voltages or branch voltages and currents are calculated at each iteration and each timepoint. Even with sparse matrix techniques the simulation of modern large-scale integrated (LSI) circuits is not possible in many situations due to the excessive computation time and high storage requirements. The latency approach is a circuit analysis version of the selective trace approach used in logic simulation. This approach takes advantage of the fact that in some circuits only a small portion of the circuit is active at any given time and at any iteration, and thus provides savings in CPU time.

In the program SLATE this approach is applied at three levels: (1) device level; (2) subnetwork level; (3) network level. At the device level, it is also called the bypass scheme. This scheme is done by monitoring the operating point of each nonlinear device. If the operating point does not change significantly between timepoints or Newton-Raphson iterations, then the device models are not reevaluated, and the matrix entries computed at the previous timepoint or the previous iteration are used again. This scheme is used in SPICE2 [2] and SPLICE [39]. Latency at the subnetwork and network levels can be well exploited when tearing methods are used to analyze the network. The tearing method used in program SLATE is node tearing, so in the following discussion about latency at the subnetwork and network levels, node tearing is assumed. Latency exploitation at the subnetwork level is presented in Section 6.1. Latency exploitation at the network level is presented in Section 6.2. In Section 6.3 a discussion is given.

6.1. Latency Exploitation at the Subnetwork Level

As discussed in Chapter V, the node tearing method is just a generalized Norton equivalent circuit approach. After all the internal circuit variables of a subnetwork are eliminated, a generalized Norton equivalent circuit of the subnetwork is obtained. Combining the equivalent circuits of all the subnetworks with the rest of the network (Fig. 6.1), we obtain the interconnection circuit. So after applying the node tearing method to tear the network apart into several subnetworks, the preprocessing of each subnetwork to obtain the contribution to the interconnection matrix is equivalent to constructing an equivalent circuit for each subnetwork. If the solutions of the circuit variables of a subnetwork do not change significantly between timepoints or Newton-Raphson iterations, then there is no need to reconstruct an equivalent circuit for that subnetwork. The equivalent circuit constructed at the previous timepoint or the previous iteration is used again and the subnetwork is declared as latent. The subnetwork remains latent until the solutions of the circuit variables of the subnetwork change significantly between when it is declared latent and the present time or the present iteration. This is the basic concept of latency at subnetwork level.

There are two types of latency at the subnetwork level: one is latency in the Newton-Raphson iteration, the other is latency in time.

Latency in the Newton-Raphson iteration is not natural. It is related to the convergence property of each subnetwork and the initial guess of the operating point for each subnetwork. Let us consider the example in Fig. 6.2. It is assumed that the input signal is constant and that a dc analysis is required. Different subnetworks may require different number

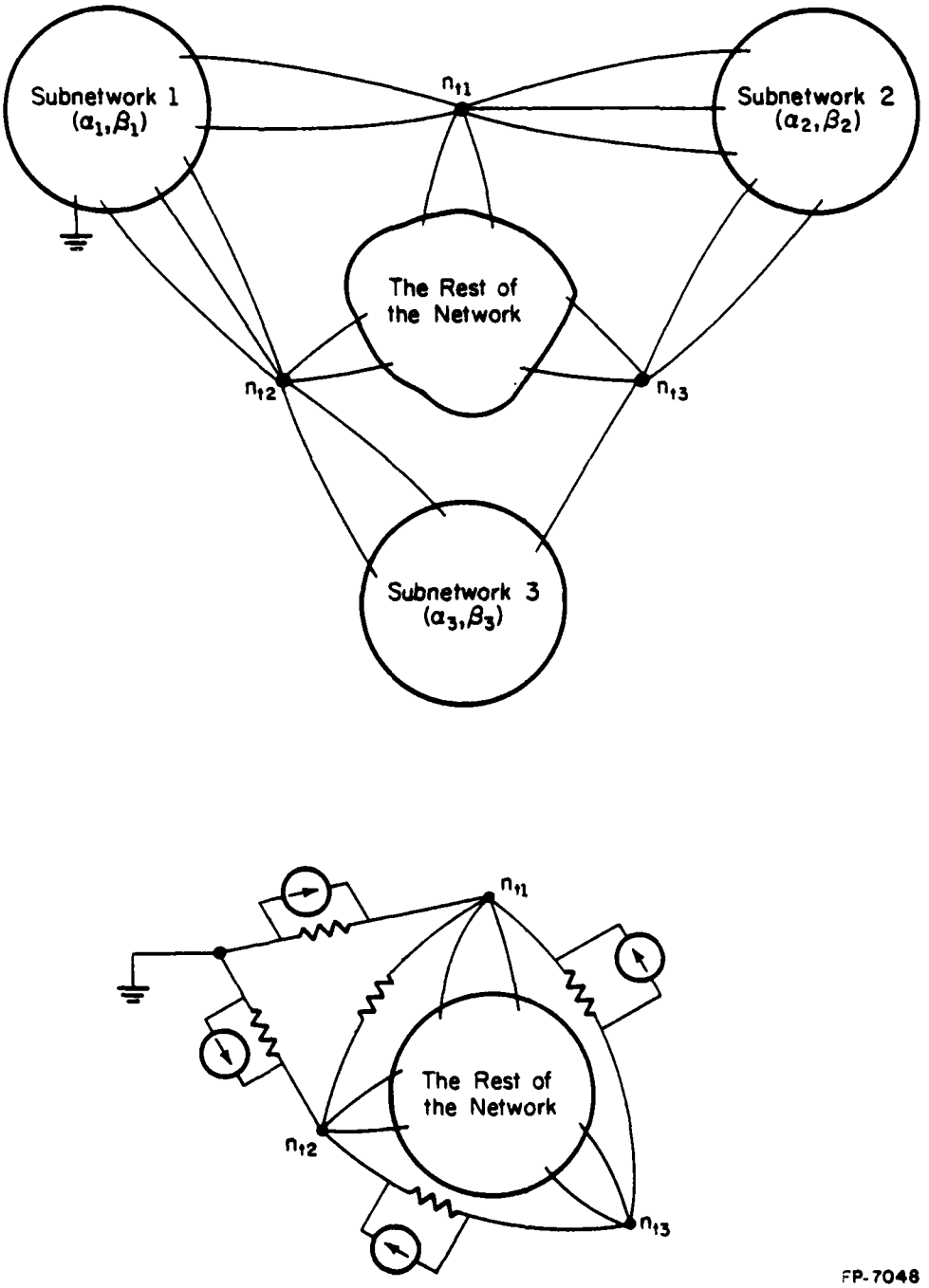


Fig. 6.1(a) A Network Partitioned into Three Subnetworks by the Node Tearing Method.

(b) Equivalent Interconnection Circuit obtained from the Node Tearing Method.

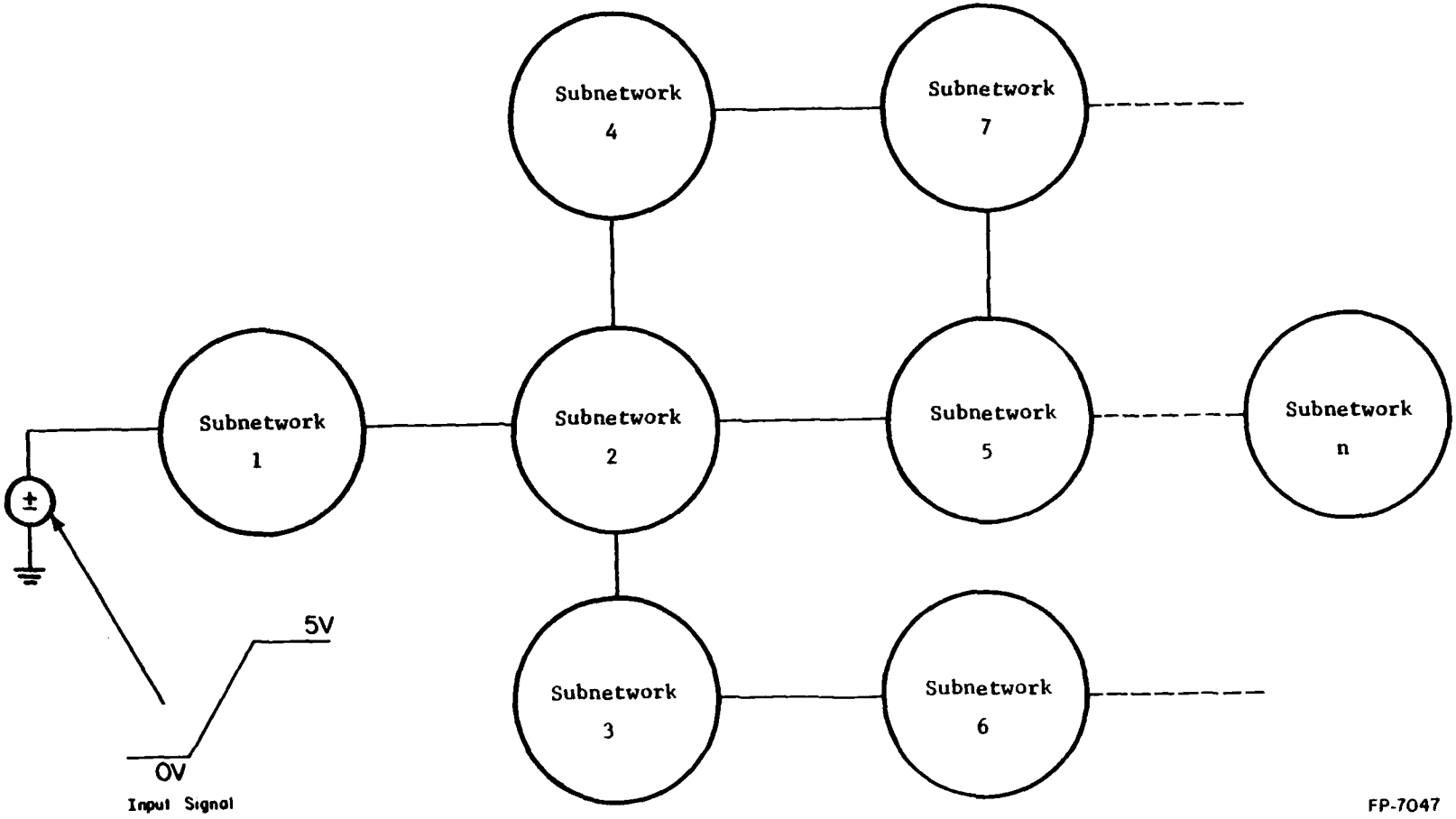


Fig. 6.2 Example of Latency.

of iterations to converge, for example, because the initial guess of the operating points may be good for some subnetworks and bad for the others. After these subnetworks converge, they are declared as latent. The Newton-Raphson iterations continue until all the subnetworks converge. This phenomenon is called latency in the Newton-Raphson iteration. Taking advantage of this latency results in savings in the execution time. This latency may not exist in some circuits. For example, a linear circuit.

Latency in time is a natural phenomenon. All physical devices have intrinsic delay time between excitation and response. For a large network, when the input signal changes, it takes time for this change to propagate to the rest of the network. Let us consider the example in Fig. 6.2 again. Let us assume that the input changes from 0V to 5V at time t_0. Initially, probabably only the first few subnetworks are not latent, the rest of the subnetworks are latent. As time passes, the change in the response propagates to the intermediate subnetworks. At this time, only the intermediate subnetworks are not latent, the rest of the subnetworks are latent. Finally, the change propagates to the last few subnetworks and the rest of the subnetworks are latent. This phenomenon is called latency in time, and it always exists in real circuits. Taking advantage of this latency results in savings in the execution time.

In order to exploit these two types of latency, some sort of latency criteria are required to determine if a subnetwork is latent. The latency criteria proposed here are developed for the node tearing method, if the branch tearing method is used, these criteria should be modified accordingly.

Let us consider one subnetwork N_k. Let the tearing node voltages of N_k be denoted by v_{tk}, the node voltages of N_k be denoted by v_{sk}, the node voltages of all the nonlinear devices be denoted by v_{nlk}.

First, let us consider the latency criterion in the Newton-Raphson iteration. In principle, the solution of all the circuit variables of a subnetwork should be checked to determine if the subnetwork is latent. However, to check for latency in the Newton-Raphson iteration only the node voltages of all the nonlinear devices need be checked. The latency criterion in the Newton-Raphson iteration used in SLATE is as follows: A subnetwork N_k is declared as latent at the ith iteration if the following two conditions are satisfied.

$$(1)\quad |v_{nlk_m}(i-1)-v_{nlk_m}(i-2)| \leq \varepsilon_a + \varepsilon_r \max(|v_{nlk_m}(i-1)|, \quad (6.1)$$

$$|v_{nlk_m}(i-2)|) \qquad m = 1,2,....$$

$$(2)\quad |v_{tk_m}(i)-v_{tk_m}(i-1)| \leq \varepsilon_a + \varepsilon_r \max(|v_{tk_m}(i)|, |v_{tk_m}(i-1)|) \quad (6.2)$$

$$m = 1,2,....$$

where ε_a and ε_r are absolute and relative error criteria.

The subnetwork N_k will remain latent as long as

$$(3)\quad |v_{tk_m}(i+j)-v_{tk_m}(i-1)| \leq \varepsilon_a + \varepsilon_r \max(|v_{tk_m}(i+j)|, \quad (6.3)$$

$$|v_{tk_m}(i-1)|) \qquad m = 1,2,....$$
$$j = 1,2,....$$

Once a subnetwork is declared as latent in the Newton-Raphson iteration, no linearization of the nonlinear devices of N_k is required, no preprocessing of the subnetwork to obtain the partial contribution to the interconnection matrix is required, no backward substitution to obtain the solutions of the internal circuit variables is required, and no convergence

tests are required. One only needs to monitor the tearing node voltages and to bring the previous partial contribution to the interconnection matrix.

The simulation data from SLATE show that considerable savings in execution time is obtained and the output results are essentially the same as those from YSPICE. Table 6.1 gives the simulation data from SLATE for the circuit shown in Fig. 6.3. A dc analysis was performed. For this circuit, a 42.42% latency exploitation was achieved and a 22.65% savings in CPU time was obtained.

Remark: Because the CPU time shown in Table 6.1 is the total CPU time, which includes the time spent in the I/O and other utility subroutines, the savings in CPU time is not the same as the latency exploitation.

For the latency in time criteria, four schemes are proposed here. The first three schemes, scheme 0, scheme 1 and scheme 2, have been implemented and tested in program SLATE. Scheme 3 is still under investigation.

Scheme 0 is the easiest and the crudest scheme that could be implemented. A subnetwork N_k is considered latent at time t_n if

$$(1)\quad |v_{tk_m}(t_n)-v_{tk_m}(t_{n-1})| \leq \varepsilon_a+\varepsilon_r \max(|v_{tk_m}(t_n)|, |v_{tk_m}(t_{n-1})|) \qquad m = 1,2,.... \tag{6.4}$$

The subnetwork N_k will remain latent as long as

$$(2)\quad |v_{tk_m}(t_{n+j})-v_{tk_m}(t_{n-1})| \leq \varepsilon_a+\varepsilon_r \max(|v_{tk_m}(t_{n+j})|, |v_{tk_m}(t_{n-1})|) \qquad \begin{matrix} m = 1,2,.... \\ j = 1,2,.... \end{matrix} \tag{6.5}$$

The advantages of Scheme 0 are:

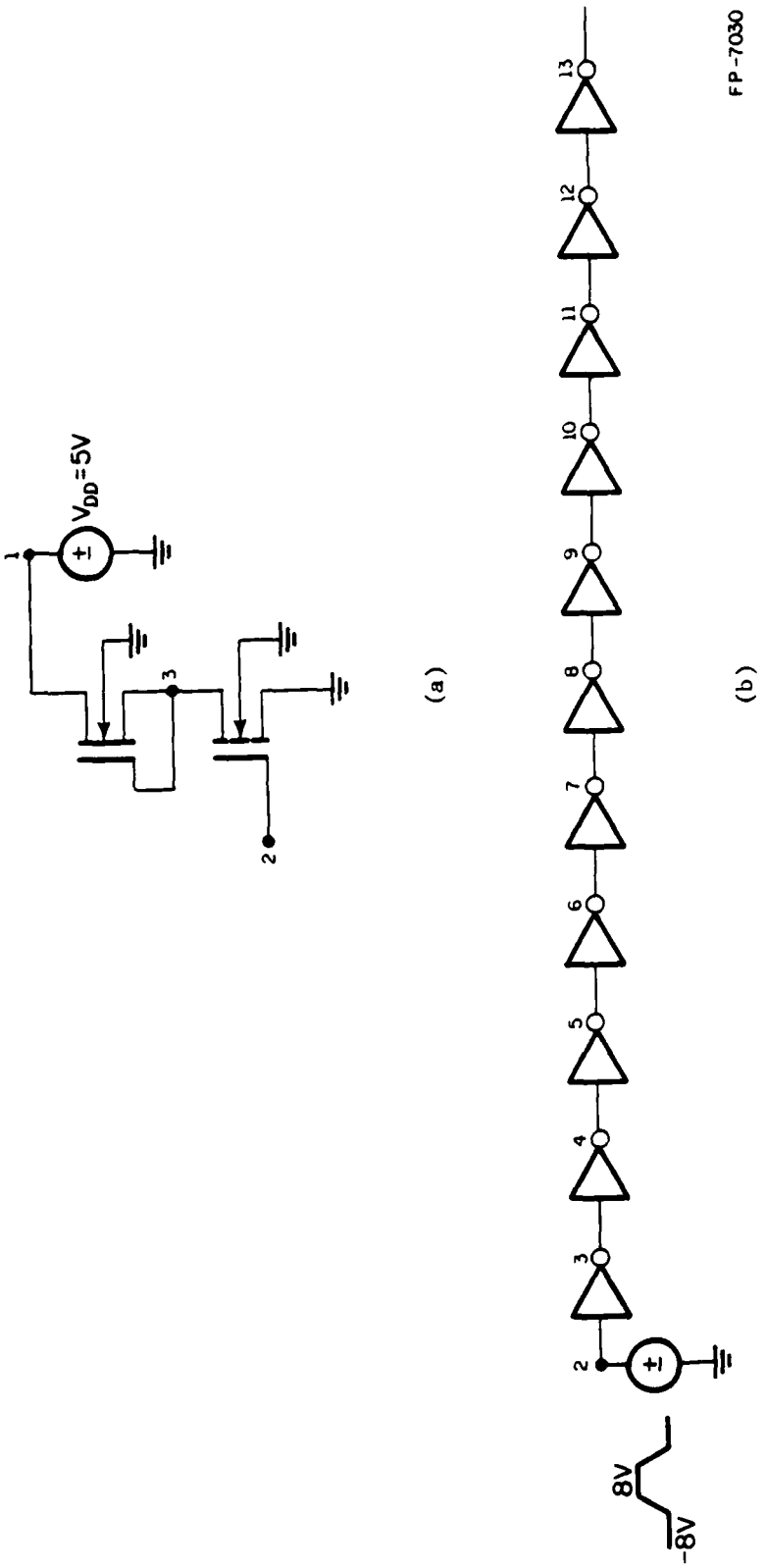


Fig. 6.3(a) Subcircuit: An MOS Inverter Gate.
(b) Entire Network: A Chain of Inverters.

Table 6.1 Simulation Data of a DC Analysis for the MOS Circuit in Fig. 6.3.

DC analysis	SLATE	SPICE2
# of subnetworks times # of iterations	231	231
# of nonlatent subnetworks times # of iterations	133	
Latency exploitation	42.42%	
Total CPU time (sec.)	3.927	5.077
Savings in CPU time	22.65%	

(1) It is very easy to implement and there is no overhead.

(2) It is faster than Scheme 1

The disadvantages of Scheme 0 are:

(1) It is not reliable and it is not accurate. If the network is a stiff system and some of the node voltages are slowly varying, then it is possible to declare a slowly varying subnetwork as latent and consequently wrong answers are obtained.

(2) The tearing node voltages at time t_{n-1} must be stored, that is , more memory is required.

Scheme 1 is the most accurate scheme, and it only takes advantage of latency in the Newton-Raphson iteration. It is based on the idea that if a subnetwork is latent in time, then it is also latent in the Newton-Raphson iteration. Even if we do not take advantage of latency in time, all the subnetworks are treated as nonlatent in time and are solved at least once at every timepoint. Those subnetworks which are latent in time at any timepoint will be declared as latent in the Newton-Raphson iteration after one iteration at that timepoint. So at most one iteration for each latent subnetwork is wasted at one timepoint. Scheme 1 is as follows:

(1) Solve the entire network including all the subnetworks at least once at every timepoint.

(2) If any subnetwork is latent in time, then it is latent in all the subsequent Newton-Raphson iterations at that timepoint. So only one iteration for that subnetwork is performed.

(3) For the nonlatent subnetworks, the Newton-Raphson iterations are continued until convergence is obtained at that timepoint.

The advantages of Scheme 1 are:

(1) It is very easy to implement and there is no overhead.

(2) The tearing node voltages at time t_{n-1} need not be stored, that is, less memory is required.

(3) It is accurate and reliable.

The disadvantage of Scheme 1 is that at every timepoint one iteration is wasted for each subnetwork latent in time.

Scheme 0 is efficient but is not reliable. Scheme 1 is reliable but is not efficient. If the network being solved is not stiff, then Scheme 0 is preferred. However, if the network is stiff and efficiency is important, then both Scheme 0 and Scheme 1 are not suitable. Scheme 2 was developed to accommodate this situation. It is similar to Scheme 0 in efficiency and it is similar to Scheme 1 in reliability. It differs from Scheme 0 in that some extra checks are made to make sure that slowly varying subnetworks will not be declared as latent. All the slowly varying subnetworks are declared as nonlatent.

Let the charges of capacitors and the fluxes of inductors of subnetwork N_k be denoted by $\underset{\sim}{Q}_k = (Q_{k1}, Q_{k2}, \ldots\ldots, Q_{kb})$. Let the currents of capacitors and the voltages of inductors of subnetwork N_k be denoted by $\underset{\sim}{I}_k = (I_{k1}, I_{k2}, \ldots\ldots, I_{kb})$. Scheme 2 is as follows: A subnetwork N_k is considered as latent if the following three conditions are satisfied.

$$(1)\quad |v_{tk_m}(t_n)-v_{tk_m}(t_{n-1})| \leq \varepsilon_a+\varepsilon_r \max(|v_{tk_m}(t_n)|, \quad (6.6)$$
$$|v_{tk_m}(t_{n-1}) \qquad m = 1, 2, \ldots,$$

This condition is the same as that of Scheme 0.

$$(2)\quad |I_{k_m}(t_n)-I_{k_m}(t_{n-1})| \leq \varepsilon_c+\varepsilon_r \max(|I_{k_m}(t_n)|, \quad (6.7)$$
$$|I_{k_m}(t_{n-1})|) \qquad m = 1,2 \ldots, b$$

where ε_c is the absolute error criterion for current. This condition is used to check if the changes of the energy-storage elements of subnetwork N_k are small.

$$(3)\quad h_{n-1}\left|\frac{I_{k_m}(t_n)-I_{k_m}(t_{n-1})}{Q_{k_m}(t_n)-Q_{k_m}(t_{n-1})}\right| \geq 1 \qquad m = 1, 2, \ldots, b \quad (6.8)$$

This condition is used to check if there are slowly varying nodes within subnetwork N_k. In order to avoid division by zero, if $|I_{k_m}(t_n)-I_{k_m}(t_{n-1})| \leq \varepsilon$ (ε is a very small quantity, it is 10^{-12} in SLATE), then condition (3) is skipped.

The subnetwork N_k will remain latent as long as

$$(4)\quad |v_{tk_m}(t_{n+j})-v_{tk_m}(t_{n-1})| \leq \varepsilon_a+\varepsilon_r \max(|v_{tk_m}(t_{n+j})|, \quad (6.9)$$
$$|v_{tk_m}(t_{n-1})|) \qquad m = 1, 2, \ldots$$

Eq. (6.8) is derived based on the following reasoning. Let us assume that we are dealing with a linear capacitor and an exponential waveform, then

$$Q = C*V \quad (6.10)$$

$$\frac{dQ}{dt} = I \quad (6.11)$$

$$V = V_{DD}(1 - e^{-t/\tau}) \tag{6.12}$$

$$Q(t_n) = C*V_{DD}(1 - e^{t_n/\tau}) \tag{6.13}$$

$$Q(t_{n-1}) = C*V_{DD}(1 - e^{t_{n-1}/\tau}) \tag{6.14}$$

$$I(t_n) = -\frac{C*V_{DD}}{\tau} e^{-t_n/\tau} \tag{6.15}$$

$$I(t_{n-1}) = -\frac{C*V_{DD}}{\tau} e^{-t_{n-1}/\tau} \tag{6.16}$$

From Eqs. (6.13), (6.14), (6.15), and (6.16), we obtain

$$h_{n-1}\frac{I(t_n)-I(t_{n-1})}{Q(t_n)-Q(t_{n-1})} = \frac{h_{n-1}}{\tau} \tag{6.17}$$

So Eq. (6.8) means that although the change in the response of a capacitor is very small, if h_{n-1} is smaller than τ, then the capacitor is not latent, it is just slowly varying.

The advantages of Scheme 2 are:

(1) It is faster than Scheme 1.

(2) It is accurate and reliable. Slowly varying subnetworks are detected and are treated as nonlatent subnetworks.

The disadvantages of Scheme 2 are:

(1) More checking is required.

(2) The tearing node voltages at time t_{n-1} must be stored, that is, more memory is required.

(3) slowly varying subnetworks are detected and are treated as nonlatent subnetworks, even though the changes in the response may be negligible.

Remark: The detection of slowly varying subnetworks increases the accuracy and reliability of the program, that is why it is an advantage. However, since the changes in these slowly varying subnetworks are small, treating them as nonlatent subnetworks is not efficient, that is why it is also a disadvantage.

In order to overcome this problem, Scheme 3 was proposed. Scheme 3 takes full advantage of latency in time. Scheme 3 is as follows:

(1) The truncation error criteria are used to determine the timestep for each subnetwork.

(2) Each subnetwork is analyzed with its own timestep.

The advantages of Scheme 3 are:

(1) It should be faster than all the other schemes.

(2) It is accurate and reliable. Since every subnetwork has its own timestep, there will not be the problem of slowly varying subnetworks.

The disadvantages of Scheme 3 are:

(1) It is not compatible with the present version of SLATE. An extra event scheduler is required and the data structures of SLATE has to be revised.

(2) It is more complicated to implement.

Because Scheme 3 is not compatible with the present version of SLATE, therefore it has not been tested and implemented in SLATE.

In the following, a small selection of examples is presented to give a comparison among the first three schemes of SLATE: Scheme 0, Scheme 1, and Scheme 2, and YSPICE.

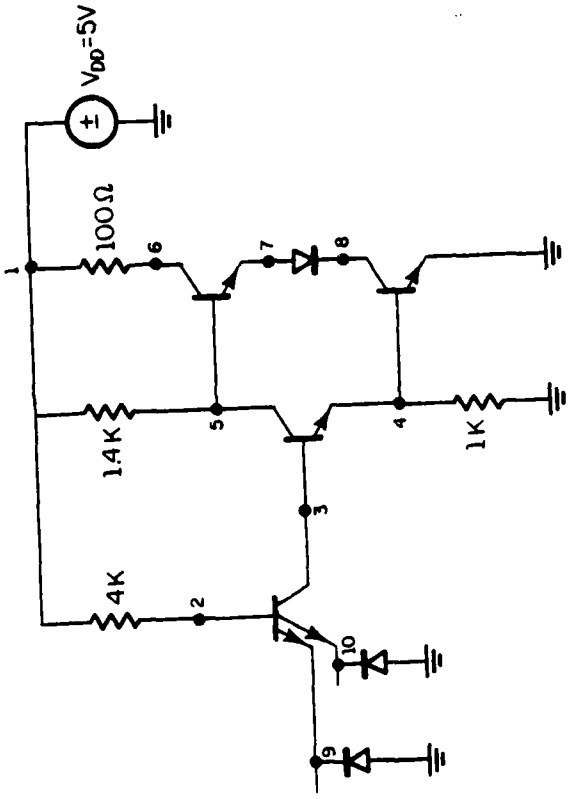
Example 6.1: The MOS circuit shown in Fig 6.3 was analyzed by SLATE and YSPICE. The output results of the three schemes of SLATE and YSPICE are essentially the same (within four significant figures). The simulation data of a transient analysis are given in Table 6.2. The simulation data show that both Scheme 0 and Scheme 2 are more efficient than Scheme 1. For this circuit, Scheme 2 is the most efficient and is about 2.5 times faster than YSPICE.

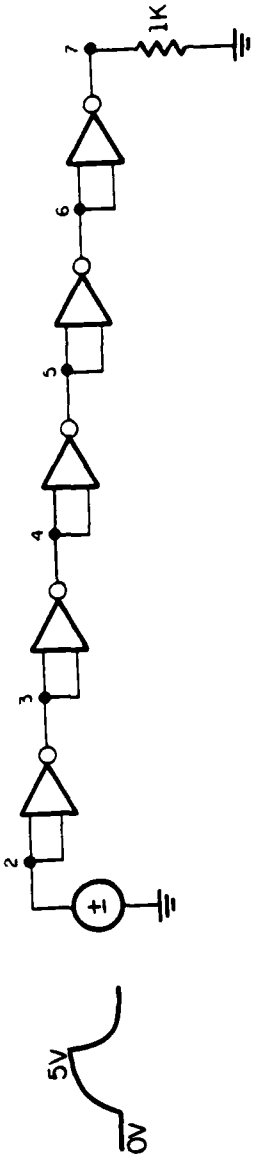
This example shows that the latency in time approach is useful for the analysis of MOS circuits. The next example shows that the latency in time approach is also useful for bipolar circuits.

Example 6.2: The TTL circuit shown in Fig 6.4 was analyzed by SLATE and YSPICE. The output results of the three schemes of SLATE and YSPICE are essentially the same(within four significant figures). The simulation data of a transient analysis are given in Table 6.3. For this example, the

Table 6.2 Simulation Data of a Transient Analysis for the MOS Circuit in Fig. 6.3.

Transient analysis	Scheme 0	Scheme 1	Scheme 2	YSPICE
# of subnetworks times # of iterations	2849	2475	2585	
# of nonlatent subnetworks times # of iterations	790	1277	706	
Latency exploitation	72.27%	48.40%	72.69%	
Total CPU time (sec.)	18.358	22.073	17.022	42.877
Savings in CPU time	57.12%	48.52%	60.30%	





FP-7031

Fig. 6.4(a) Subcircuit: A TTL NAND Gate.
(b) Entire Network: A Chain of Inverters.

Table 6.3 Simulation Data of a Transient Analysis for the TTL Circuit in Fig. 6.4.

Transient analysis	Scheme 0	Scheme 1	Scheme 2	YSPICE
# of subnetworks times # of iterations	2850	2945	2770	
# of nonlatent subnetworks times # of iterations	1516	2128	1945	
Latency exploitation	46.81%	27.74%	29.78%	
Total CPU time (sec.)	68.996	97.164	86.033	132.338
Savings in CPU time	47.86%	26.58%	34.99%	

simulation data show that Scheme 0 is the most efficient, Scheme 1 is still the least efficient. The reason why Scheme 2 is not as efficient as Scheme 0 can be traced to the close-coupling of bipolar circuits. So the conclusion obtained from this example is that the error criteria should be loosened for bipolar circuits. Scheme 0 is about 2 times faster than YSPICE.

Although the above two examples show that Scheme 0 is very efficient, however, as mentioned before, Scheme 0 has a reliability problem. The following example shows that Scheme 0 may give inaccurate output results for stiff systems.

Example 6.3: The RC circuit shown in Fig. 6.5 was analyzed by the three schemes of SLATE. This circuit is a stiff system. the output results are given in Table 6.4. For scheme 0, because the changes of the tearing node voltages of subnetwork 1 and subnetwork 2 are very small after t = 13 ns, both subnetworks are declared as latent. Since the input is constant too after t = 13 ns, all the calculated output voltages will remain unchanged afterwards, while the true output voltages should increase slowly. This phenomenon can be observed from the unchanged output voltages after t = 13 ns in Table 6.4(a). This example shows that Scheme 0 may not give accurate results when the network is stiff and that both Scheme 1 and Scheme 2 give accurate results even when the network is stiff.

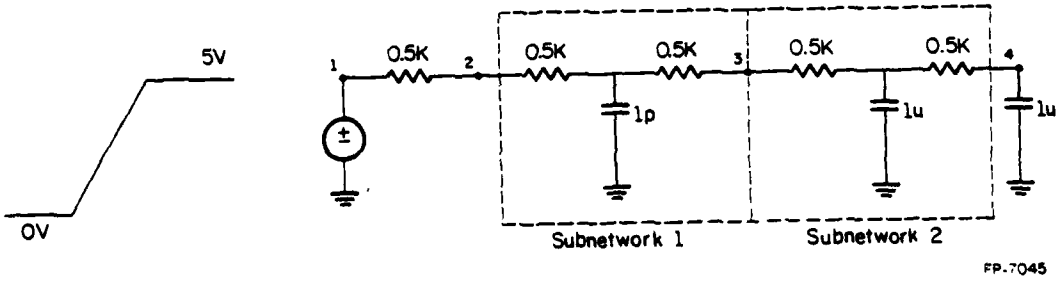


Fig. 6.5 An Example of a Stiff System.

Table 6.4(a) Scheme 0.

TIME	V(2)	V(3)	V(4)
0.000D+00	0.000D+00	0.000D+00	0.000D+00
1.000D-09	2.936D+00	7.567D-01	4.445D-13
2.000D-09	3.604D+00	1.146D+00	3.726D-12
3.000D-09	3.733D+00	1.237D+00	1.144D-11
4.000D-09	3.749D+00	1.249D+00	2.403D-11
5.000D-09	3.751D+00	1.251D+00	4.161D-11
6.000D-09	3.749D+00	1.249D+00	6.187D-11
7.000D-09	3.750D+00	1.250D+00	8.020D-11
8.000D-09	3.750D+00	1.250D+00	9.850D-11
9.000D-09	3.749D+00	1.249D+00	1.161D-10
1.000D-08	3.749D+00	1.249D+00	1.260D-10
1.100D-08	3.748D+00	1.248D+00	1.307D-10
1.200D-08	3.748D+00	1.248D+00	1.302D-10
1.300D-08	3.749D+00	1.249D+00	1.246D-10
1.400D-08	3.749D+00	1.249D+00	1.145D-10
1.500D-08	3.749D+00	1.249D+00	1.145D-10
1.600D-08	3.749D+00	1.249D+00	1.145D-10
1.700D-08	3.749D+00	1.249D+00	1.145D-10
1.800D-08	3.749D+00	1.249D+00	1.145D-10
1.900D-08	3.749D+00	1.249D+00	1.145D-10
2.000D-08	3.749D+00	1.249D+00	1.145D-10
2.100D-08	3.749D+00	1.249D+00	1.145D-10
2.200D-08	3.749D+00	1.249D+00	1.145D-10
2.300D-08	3.749D+00	1.249D+00	1.145D-10
2.400D-08	3.749D+00	1.249D+00	1.145D-10
2.500D-08	3.749D+00	1.249D+00	1.145D-10

Table 6.4(b) Scheme 1.

TIME	V(2)	V(3)	V(4)
0.000D+00	0.000D+00	0.000D+00	0.000D+00
1.000D-09	2.936D+00	7.567D-01	4.444D-13
2.000D-09	3.604D+00	1.146D+00	3.726D-12
3.000D-09	3.733D+00	1.237D+00	1.144D-11
4.000D-09	3.749D+00	1.249D+00	2.403D-11
5.000D-09	3.751D+00	1.251D+00	4.161D-11
6.000D-09	3.749D+00	1.249D+00	6.187D-11
7.000D-09	3.750D+00	1.250D+00	8.020D-11
8.000D-09	3.750D+00	1.250D+00	9.850D-11
9.000D-09	3.749D+00	1.249D+00	1.172D-10
1.000D-08	3.749D+00	1.249D+00	1.404D-10
1.100D-08	3.749D+00	1.249D+00	1.666D-10
1.200D-08	3.749D+00	1.249D+00	1.958D-10
1.300D-08	3.750D+00	1.250D+00	2.281D-10
1.400D-08	3.751D+00	1.251D+00	2.629D-10
1.500D-08	3.751D+00	1.251D+00	2.919D-10
1.600D-08	3.751D+00	1.251D+00	3.208D-10
1.700D-08	3.751D+00	1.251D+00	3.498D-10
1.800D-08	3.751D+00	1.251D+00	3.788D-10
1.900D-08	3.751D+00	1.251D+00	4.077D-10
2.000D-08	3.751D+00	1.251D+00	4.367D-10
2.100D-08	3.751D+00	1.251D+00	4.656D-10
2.200D-08	3.751D+00	1.251D+00	4.946D-10
2.300D-08	3.750D+00	1.250D+00	5.236D-10
2.400D-08	3.750D+00	1.250D+00	5.525D-10
2.500D-08	3.750D+00	1.250D+00	5.815D-10

Table 6.4(c) Scheme 2.

TIME	V(2)	V(3)	V(4)
0.000D+00	0.000D+00	0.000D+00	0.000D+00
1.000D-09	2.936D+00	7.567D-01	4.444D-13
2.000D-09	3.604D+00	1.146D+00	3.726D-12
3.000D-09	3.733D+00	1.237D+00	1.144D-11
4.000D-09	3.749D+00	1.249D+00	2.403D-11
5.000D-09	3.751D+00	1.251D+00	4.161D-11
6.000D-09	3.749D+00	1.249D+00	6.187D-11
7.000D-09	3.750D+00	1.250D+00	8.020D-11
8.000D-09	3.750D+00	1.250D+00	9.850D-11
9.000D-09	3.749D+00	1.249D+00	1.172D-10
1.000D-08	3.749D+00	1.249D+00	1.404D-10
1.100D-08	3.749D+00	1.249D+00	1.666D-10
1.200D-08	3.749D+00	1.249D+00	1.958D-10
1.300D-08	3.750D+00	1.250D+00	2.281D-10
1.400D-08	3.751D+00	1.251D+00	2.629D-10
1.500D-08	3.751D+00	1.251D+00	2.919D-10
1.600D-08	3.751D+00	1.251D+00	3.208D-10
1.700D-08	3.751D+00	1.251D+00	3.498D-10
1.800D-08	3.751D+00	1.251D+00	3.788D-10
1.900D-08	3.751D+00	1.251D+00	4.077D-10
2.000D-08	3.751D+00	1.251D+00	4.367D-10
2.100D-08	3.751D+00	1.251D+00	4.656D-10
2.200D-08	3.751D+00	1.251D+00	4.946D-10
2.300D-08	3.750D+00	1.250D+00	5.236D-10
2.400D-08	3.750D+00	1.250D+00	5.525D-10
2.500D-08	3.750D+00	1.250D+00	5.815D-10

6.2. Latency Exploitation at the Network Level

When the submatrices are large and the interconnection matrix is small and sparse, then latency exploitation at the subnetwork level provides most of the savings in CPU time. When the reverse is true, that is, the submatrices are small and the interconnection matrix is large and relatively dense, then the latency exploitation at network level becomes important. Usually, the latter situation is true for MOS circuits.

Latency exploitation at the network level is equivalent to solving a smaller interconnection matrix by using voltage source substitution. From the substitution theorem we know that the same results will be obtained. This approach can be explained by the following example as shown in Fig. 6.6(a). Let us assume that at a particular time or a particular iteration, only subnetworks 5 and 6 are nonlatent, all the other subnetworks are latent. By using voltage source substitution, the network can be replaced by the equivalent network as shown in Fig. 6.6(b). The equivalent network is solved to obtain the solutions of all the nonlatent nodes (nodes which belong to nonlatent subnetworks). This equivalent network is obtained as follows. First, all the nonlatent subnetworks and all the latent subnetworks which are adjacent to the nonlatent subnetworks are included in the equivalent network; secondly, all the tearing nodes which only belong to latent subnetworks are replaced by voltage sources, the resulting network is the equivalent network. For this example, in the equivalent network, the nonlatent subnetworks are subnetworks 5 and 6, the latent subnetworks are subnetworks 4 and 7, the tearing nodes which are replaced by voltage sources are nodes 5 and 9. After the solution for all the nonlatent nodes is obtained, subnetworks 4 and 7 are checked to see if they remain latent. If the answer is yes, then the same equivalent circuit is

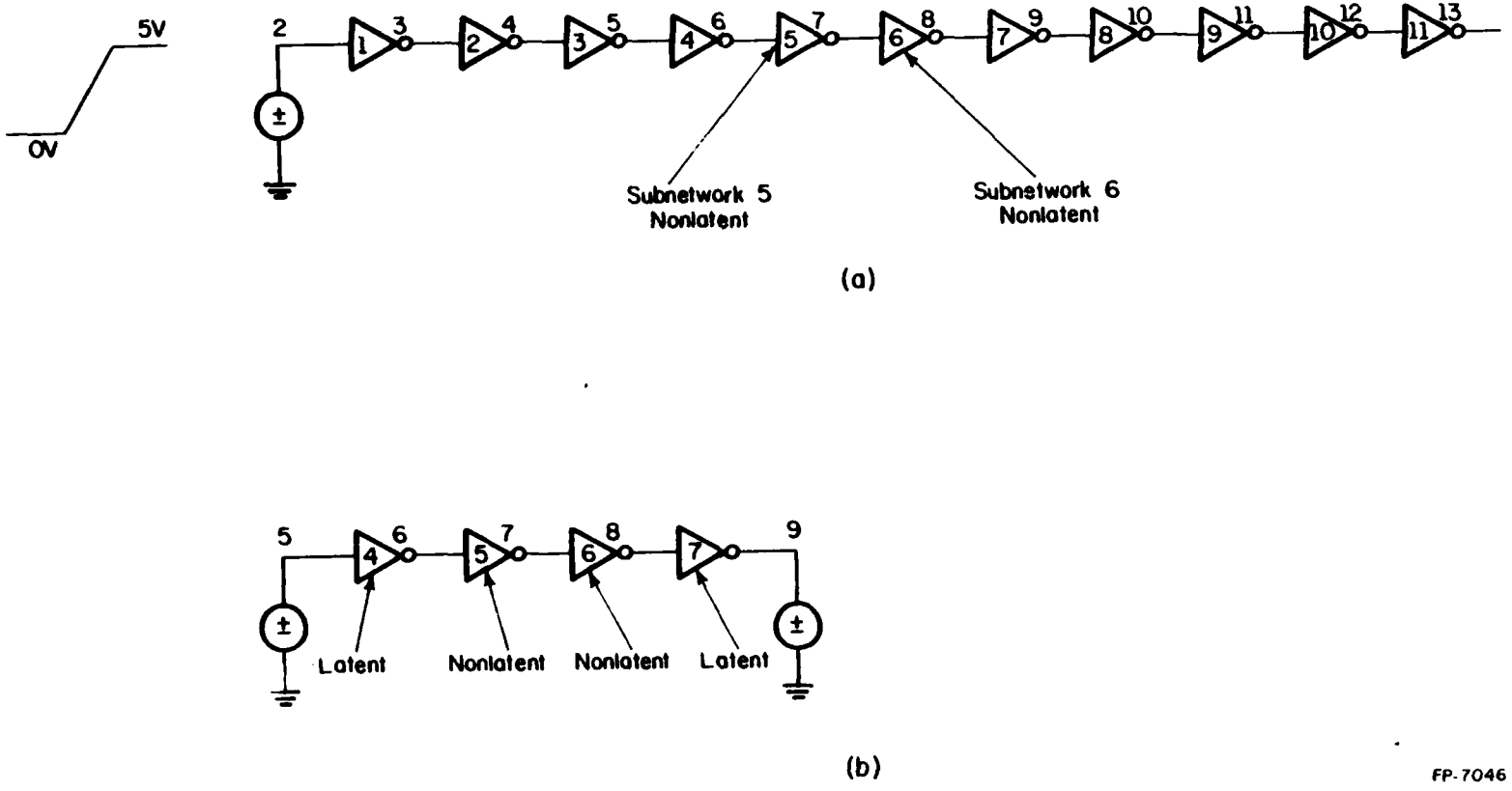


Fig. 6.6(a) A Chain of Inverters: Subnetworks 5 and 6 are nonlatent.
(b) Equivalent Network for (a).

used again. If the answer is no, then a new equivalent network is generated.

The above is the conceptual idea. In the implementation, because sparse matrix techniques are used, we do not want to really generate the equivalent network and we do not want to reorder the interconnection matrix and reconstruct the sparse matrix pointer systems everytime a new equivalent circuit is generated. So the following algorithm is implemented in SLATE.

(1) The ordering of the interconnection matrix is determined in the preprocessing phase assuming all the subnetworks are nonlatent, and this ordering is used in the whole analysis.

(2) At every timepoint or iteration, all the subnetworks are checked to determine their latent status. All the tearing nodes which only belong to latent subnetworks are labelled as latent nodes.

(3) All the rows corresponding to the latent nodes are replaced by the branch constraint relations of grounded voltage sources. This is done by skipping those rows and columns during the LU factorization and forward and backward substitutions.

Example 6.4: The MOS circuit shown in Fig. 3.18 was analyzed by SLATE with and without the latency exploitation at network level. Scheme 2 was used. The output results are essentially the same for both approaches(whithin four significant figures). The simulation data for both approaches are given in Table 6.5. This example shows that the latency exploitation at network level also provides savings in CPU time.

Table 6.5 Simulation Data of a Transient Analysis for the MOS Circuit in Fig. 3.18.

Transient analysis	Scheme 2 with latency exploitation at network level	Scheme 2 without latency exploitation at network level
Total CPU time (sec.)	80.102	102.365
Savings in CPU time	21.75%	

6.3. Discussion

Four schemes for latency exploitation at subnetwork level are proposed in this chapter. Scheme 0, Scheme 1 and Scheme 2 were implemented and tested in the program SLATE. Scheme 3 is not compatible with the p ent version of SLATE, so it is still at the development stage. From the simulation data obtained from the first three schemes, our conclusion is that Scheme 2 is the best of these three schemes. However, for bipolar circuits, Scheme 2 is not the most efficient one. Conceptually, Scheme 3 should be the optimal one, so more work will be devoted to study this scheme.

In order to illustrate the ideas and to estimate the inherent latency easily, chains of inverters are used as example circuits in this chapter. More complicated circuits are used in the next chapter to evaluate the latency approaches used in SLATE.

VII. CONCLUSIONS

The examples used in Chapter 6 are chains of inverters. One example has eleven levels of inverters, the other has five levels of inverters. From the simulation data we can see that the latency exploitation increases with the number of levels of logic gates. Since the number of levels of logic gates for those circuits is large and those circuits have very simple interconnection networks, so significant latency exploitation was obtained. In this chapter, the simulation data for some circuits, which have a complicated interconnection network and for which the number of levels of logic gates is small, are presented to see if the latency approach can provide significant savings in CPU time for these circuits. The simulation data are compared with those obtained from our DEC-10 version of SPICE2.

Example 7.1: The TTL circuit shown in Fig. 3.17 was analyzed by SLATE and SPICE2. Scheme 2 was used in SLATE. The output results of SLATE and SPICE2 are essentially the same (within four significant figures). The simulation data of a transient analysis are given in Table 7.1. For this bipolar circuit example, a 32.66% latency exploitation was achieved and a 40.15% savings in CPU time was obtained.

Example 7.2: The MOS circuit shown in Fig. 3.18 was analyzed by SLATE and SPICE2. Scheme 2 was used in SLATE. The output results of SLATE and SPICE2 are essentially the same (within four significant figures). The simulation data of a transient analysis are given in Table 7.2. For this MOS circuit example, a 22.53% latency exploitation was achieved and a 46.70% savings in CPU time was obtained.

Table 7.1 Simulation Data of a Transient Analysis for the TTL Circuit in Fig 3.17

Transient analysis	SLATE	SPICE2
# of subnetworks times # of iterations	6426	
# of nonlatent subnetworks times # of iterations	4327	
Latency exploitation	32.66%	
Total CPU time (sec.)	189.56	316.74
Savings in CPU time	40.15%	

Table 7.2 Simulation Data of a Transient Analysis for the MOS Circuit in Fig. 3.18.

Transient analysis	SLATE	SPICE2
# of subnetworks times # of iterations	3600	
# of nonlatent subnetworks times # of iterations	2789	
Latency exploitation	22.53%	
Total CPU time (sec.)	72.23	135.52
Savings in CPU time	46.70	

Also these simulation data show that not only the latency approach but also other new approaches implemented in SLATE provide savings in CPU time. This observation is obtained by noting that the latency exploitation is smaller than the savings in CPU time. These other new approaches which also provide savings in CPU time are the new reordering scheme for the modified nodal approach presented in Chapter 2 and the piecewise nonlinear approach presented in Chapter 3. The new reordering scheme for the modified nodal approach avoids the problem of pivoting on zero diagonal elements and decreases the number of operations at the same time. However, the efficiency provided by the new reordering scheme is problem-dependent. For example, if the circuit does not have voltage sources or inductors, then certainly no efficiency can be obtained by using our approach. The piecewise nonlinear approach is still at the experimental stage. All the examples we have simulated show that the use of the piecewise nonlinear approach hastens the convergence and improves the global convergence property of the Newton-Raphson method for bipolar and MOS circuits. However, the proof of global convergence or the conditions for global convergence for the piecewise nonlinear approach has not been obtained. Further research is needed to prove the global convergence, or to modify the approach we proposed to ensure global convergence. Also more work is needed to study if the strict piecewise nonlinear approach is efficient, and if it is not efficient, then the problem of how to use the ideas of the piecewise nonlinear approach to hasten the convergence and to improve the global convergence property of the Newton-Raphson method should be studied. The solution of the two problems of numerical integration makes the program more reliable and more accurate. This is described in Chapter 4. An equation was presented to compute the upperbound on the local truncation

error (LTE) from the maximum global error (GE_{max}) and the solution time T. The inaccuracy in the estimation of the local truncation error caused by different timesteps was resolved by introducing a new formula for the estimation. The inaccuracy in the estimation of the local truncation error caused by using the node voltages at timepoints of the previous switch interval was resolved by recognizing this situation and restarting the numerical integration from the breakpoint.

In Chapter 5, the ideas of tearing methods were detailed, the most efficient way of implementing the node tearing method was determined theoretically and experimentally, and a circuit interpretation of tearing methods was given.

In Chapter 6, four latency criteria schemes were proposed. The first three schemes: Scheme 0, Scheme 1 and Scheme 2, were implemented and tested. From the simulation data we conclude that Scheme 2 is the best out of these three. Scheme 3 is still under investigation and we think it should be the best scheme to exploit latency. More work is needed to study how to implement this scheme efficiently and reliably, and to find out if it is really the best scheme.

The nested subnetwork approach [41,42,43] is the approach which allows several levels of subnetworks and in which the latency approach is used at every level of the subnetworks. This approach may provide savings in the time spent in checking the latent status of subnetworks. Only latency exploitation at the network level is implemented in program SLATE and we believe that this checking time may be small, thus the savings in CPU time provided by the nested subnetworks approach probably is not significant. However, further investigation is needed to yield conclusive results.

Device characteristic latency and function latency are two concepts which may provide some more savings in CPU time. More investigations need to be done to exploit these two latencies.

In the present version of SLATE, a lot of information which is not needed is still stored because SLATE evolved from YSPICE. Due to this reason, although tearing methods should provide savings in memory, no comparison of memory usage was presented in this thesis.

REFERENCES

[1] L. W. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits," ERL Memo., No. ERL-M520, University of California, Berkeley, May 1975.

[2] L. W. Nagel and R. A. Rohrer, "Computer Analysis of Nonlinear Circuits, Excluding Radiation (CANCER)," IEEE J. Solid-State Circuits, Vol. SC-6, August 1971, pp. 166-182.

[3] R. A. Rohrer, L. W. Nagel, R. G. Meyer, and L. Weber, "CANCER: Computer Analysis of Nonlinear Circuits, Excluding Radiation," Proc. 1971 IEEE International Solid-State Circuits Conference, Philadelphia, February 18, 1971, pp. 124-125.

[4] F. S. Jenkins and S. P. Fan, "TIME---A Nonlinear DC and Time-Domain Circuit-Simulation Program," IEEE J. Solid-State Circuits, Vol. SC-6, August 1971, pp. 188-192.

[5] "ASTAP General Information Manual (GH20-1271-0)," IBM Corp., Mechanicsburg, Pa.

[6] H. Qassemzadeh, and T. R. Scott, "Algorithm for ASTAP---A Network Analysis Program," IEEE Trans. on Circuit Theory, Vol. CT-20, November 1973, pp. 628-634.

[7] "ECAP II Application Description Manual (GH20-0983)," IBM Corp., Mechanicsburg, Pa.

[8] F. H. Branin, G. R. Hogsett, R. L. Lunde, and L. E. Kugel, "ECAP II---A New Electronic Circuit Analysis Program," IEEE J. Solid-State Circuits, Vol. SC-6, August 1971, pp. 146-165.

[9] B. Dembart and L. Milliman, "Circus-2, A Digital Computer Program for Transient Analysis of Electronic Circuits," Harry Diamond Laboratories, Washington, D. C., July 1971.

[10] L. D. Milliman, W. A. Massena, and R. H. Dickhaut, "CIRCUS---A Digital Computer Program for Transient Analysis of Electronic Circuits," Harry Diamond Laboratory, Washington, D. C., Rep. AD-346-1, January 1967.

[11] P. R. Bryant, I. N. Hajj, S. Skelboe and M. Vlach, "WATAND Primer," University of Waterloo, Report 78-4, June 1978.

[12] G. Kron, "A Set of Principles to Interconnect the Solution of Physical Systems," Journal of Applied Physics, Vol. 24, No. 8., August 1953, pp. 965-980.

[13] G. Kron, Diakoptics---Piecewise Solution of Large-Scale Systems, Macdonald, London, 1963.

[14] H. H. Happ, Diakoptics And Networks, Academic Press, New York, 1971.

[15] F. H. Branin, "The relation between Kron's Method and the Classical methods of Network Analysis," The Matrix and Tensor Quarterly, Vol. 12, No. 3, March 1962, pp. 69-115.

[16] L. O. Chua and L. K. Chen, "Diakoptics and Generalized Hybrid Analysis," IEEE Trans. on Circuits and Systems, Vol. CAS-23, No. 12, December 1976, pp. 694-705.

[17] F. F. Wu, "Solution of Large Scale Networks by Tearing," IEEE Trans. on Circuits and Systems, Vol. CAS-23, No. 12, December 1976, pp. 706-713.

[18] L. O. Chua and L. K. Chen, "Nonlinear Diakoptics," Proc. IEEE Int. Symp. on Circuits and Systems, April 1975, pp. 373-376.

[19] K. U. Wang and T. Chao, "Diakoptics for Large Scale Nonlinear Time-Varying Networks," Proc. IEEE Int. Symp. on Circuits and Systems, April 1975, pp. 277-278.

[20] A. Sangiovanni-Vincentelli, L. K. Chen, and L. O. Chua, "A New Tearing Approach---Node Tearing Nodal Analysis," Proc. IEEE Int. Symp. on Circuits and Systems, April 1977, pp. 143-147.

[21] P. Linardis, K. G. Nichols, and E. J. Zaluska, "Network Partitioning and Latency Exploitation in Time Domain Analysis of Nonlinear Electronic Circuits," Proc. IEEE Int. Symp. on Circuits and Systems, May 1978, pp. 510-513.

[22] N. B. Rabbat and H. Y. Hsieh, "A Latent Macromodular Approach to Large-Scale Sparse Networks," IEEE Trans. on Circuits and Systems, Vol. CAS-22, No. 13, December 1976, pp. 745-752.

[23] H. Y. Hsieh and N. B. Rabbat, "Computer-Aided Design of Large Networks by Macromodular and Latent Techniques," IEEE Int. Symp. on Circuits and Systems, April 1977, pp. 688-692.

[24] N. B. Rabbat and H. Y. Hsieh, "Concepts of Latency in the Time-Domain Solution of Nonlinear Differential Equations," Proc. IEEE Int. Symp. on Circuits and Systems, May 1978, pp. 813-825.

[25] C. W. Ho, A. E. Ruehli and P. A. Brennan, "The Modified Nodal Approach to Network Analysis," IEEE Trans. on Circuits and Systems, Vol. CAS-22, pp. 504-509, June 1975.

[26] A. R. Newton and D. O. Pederson, "A Simulation Program with Large Scale Integrated Circuit Emphasis," IEEE Int. Symp. on Circuits and Systems, New York, May 1978.

[27] G. Arnout and H. J. Deman, "The use of Threshold Functions and Boolean-Controlled Network Elements for Macromodelling of LSI Circuits," IEEE J. Solid-State Circuits, vol. SC-13, pp. 326-332, June 1978.

[28] C. A. Desoer and E. S. Kuh, Basic Circuit Theory, McGraw-Hill Book Co., New York, 1969.

[29] H. M. Markowitz, "The Elimination Form of the Inverse and its Application to Linear Programming," Management Sci., Vol. 3, pp. 255-269, 1957.

[30] I. N. Hajj, P. Yang and T. N. Trick, "Avoiding Zero Pivots in the Modified Nodal Approach," IEEE Trans. on Circuits and Systems, to appear.

[31] W. J. McCalla and D. O. Pederson, "Elements of Computer-Aided Circuit Analysis," IEEE Trans. on Circuit Theory, Vol. CT-18, January 1971, pp. 14-26.

[32] R. D. Berry, "An Optimal Ordering of Electronic Circuit Equations for a Sparse Matrix Solution," IEEE Trans. on Circuit Theory, Vol. CT-18, January 1971, pp. 40-50.

[33] W. H. Kao, "Comparison of Quasi-Newton Methods for the DC analysis of Electronic Circuits," M. S. Thesis, University of Illinois at Urbana-Champaign, 1972.

[34] H. Gupta and J. Sharma, "An Algorithm for DC Solution of Electronic Circuits," IEEE Int. Symp. on Circuits and Systems, July 1979.

[35] M. E. Daniel, "Development of Mathematical Models of Semiconductor Devices for Computer-Aided Circuit Analysis," Proceeding of the IEEE, Vol. 55, No. 11, Nov. 1967.

[36] L. O. Chua and P. M. Lin, Computer-Aided Analysis of Electronic Circuits: Algorithms and Computational Techniques. Englewood Cliff, New Jersey, Prentice-Hall, 1975.

[37] F. H. Branin, "A Sparse Matrix Modification of Kron's Method of Piecewise Analysis," Proc. IEEE Int. Symp. on Circuits and Systems, April 1975, pp. 383-386.

[38] I. N. Hajj, "Sparsity Considerations in Network Solution by Tearing," IEEE Trans. on Circuits and Systems, May 1980, pp. 357-366.

[39] A. R. Newton and D. O. Pederson, "A Simulation Program with Large Scale Integrated Circuit Emphasis," IEEE Int. Symp. on Circuits and Systems, New York, May 1978.

[40] J. Katzenelson, "An Algorithm for Solving Nonlinear Resistive Network", Bell Syst. Tech. J., Vol. 44, pp. 1605-1620, Oct. 1965.

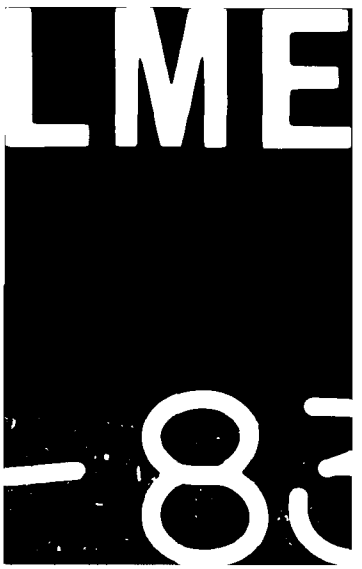
[41] A. E. Ruehli, A. L. Sangiovanni-Vincentelli and N. B. Rabbat, " Time Analysis of Large-Scale Circuits Containing One-Way Macromodels," Proc. IEEE Int. Symp. on Circuits and Systems, April 1980, pp. 766-770.

[42] N. B. Rabbat, A. L. Sangiovanni-Vincentelli and H. Y. Hsieh, "A Multilevel Newton Algorithm with Macromodeling and Latency for the Analysis of Large-Scale Nonlinear Circuits in the Time Domain", IEEE Trans. on Circuits and Systems, Vol. CAS-26, pp. 733-741, Sept. 1979.

[43] H. Y. Hsieh and N. B. Rabbat, "Multilevel Newton Algorithm for Nested Macromodel Analysis of Bipolar Networks", Proc. IEEE Int. Symp. on Circuits and Systems, April 1980, pp. 762-765.

VITA

Ping Yang was born in Ping-Tung, Taiwan, China on July 15, 1952. He attended National Taiwan University in Taipei and Graduated in June 1974 with a Bachelor of Science in Physics. He was an Ensign in the Chinese Navy from 1974 to 1976. In August 1976 he entered the University of Illinois. From August 1976 to August 1980 he held a teaching assistantship in the Electrical Engineering department and a research assistantship with the Coordinated Science Laboratory. His research interests are in the areas of computer-aided design, circuits and systems, large-scale integrated circuits, and solid-state electronics.


LME